Section 5

Floppy Disk Controller

Section 5 Contents

National Semiconductor

DP8473 Floppy Disk Controller PLUS-2™

General Description

This controller is a full featured floppy disk controller that is software compatible with the μPD765A, but also includes many additional hardware and software enhancements. These enhancements include additional logic specifically required for an IBM® PC, PC-XT®, PC-AT®, or PS/2® design.

This controller incorporates a precision analog data separator, that includes a self trimming delay line and VCO. Up to three external filters are switched automatically depending on the data rate selected. This provides optimal performance at the standard PC data rates of 250/300 kb/s, and 500 kb/s. It also enables optimum performance at 1 Mb/s (MFM). These features combine to provide the lowest possible PLL bandwidth, with the greatest lock range, and hence the widest window margin.

This controller includes write precompensation circuitry. A shift register is used to provide a fixed 125 ns early-late precompensation for all tracks at 500k/300k/250 kb/s (83 ns for 1 MB/s), or a precompensation value that scales with

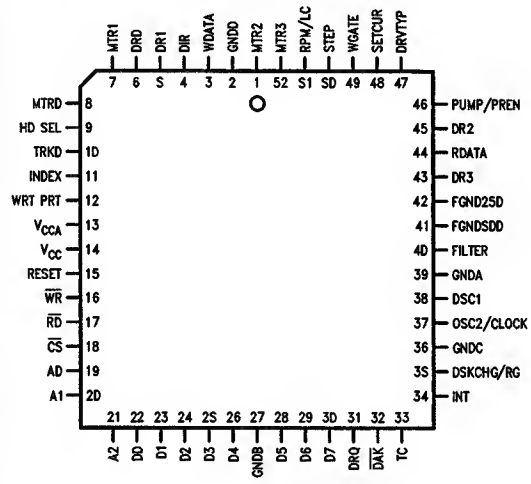
(Continued)

Features

- Fully μPD765A and IBM-BIOS compatible
- Integrates all PCXT®, PCAT®, and most PS/2® Logic
  - — On chip 24 MHz Crystal Oscillator
  - — DMA enable logic
  - — IBM compatible address decode of A0–A2
  - — 12 mA μP bus interface buffers
  - — 40 mA floppy drive interface buffers
  - — Data rate and drive control registers
- Precision analog data separator
  - — Self-calibrating PLL and delay line
  - — Automatically chooses one of three filters
  - — Intelligent read algorithm
- Two pin programmable precompensation modes
- Other enhancements
  - — up to 1 Mb/s data rate
  - — Implied seek up to 4000 tracks
  - — IBM or ISO formatting
- Low power CMOS, with power down mode

Connection Diagrams

Plastic Leaded Chip Carrier

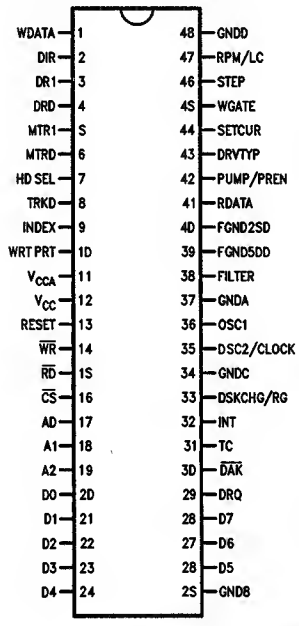


TL/F/9384–1

Top View

Order Number DP8473V
See NS Package Number V52A

Dual-In-Line Package



TL/F/9384–2

Top View

Order Number DP8473N
See NS Package Number N48A

General Description (Continued)

the data rate, 83 ns/125 ns/208 ns/ 250 ns for data rates of 1.0M/500k/300k/250 kb/s respectively.

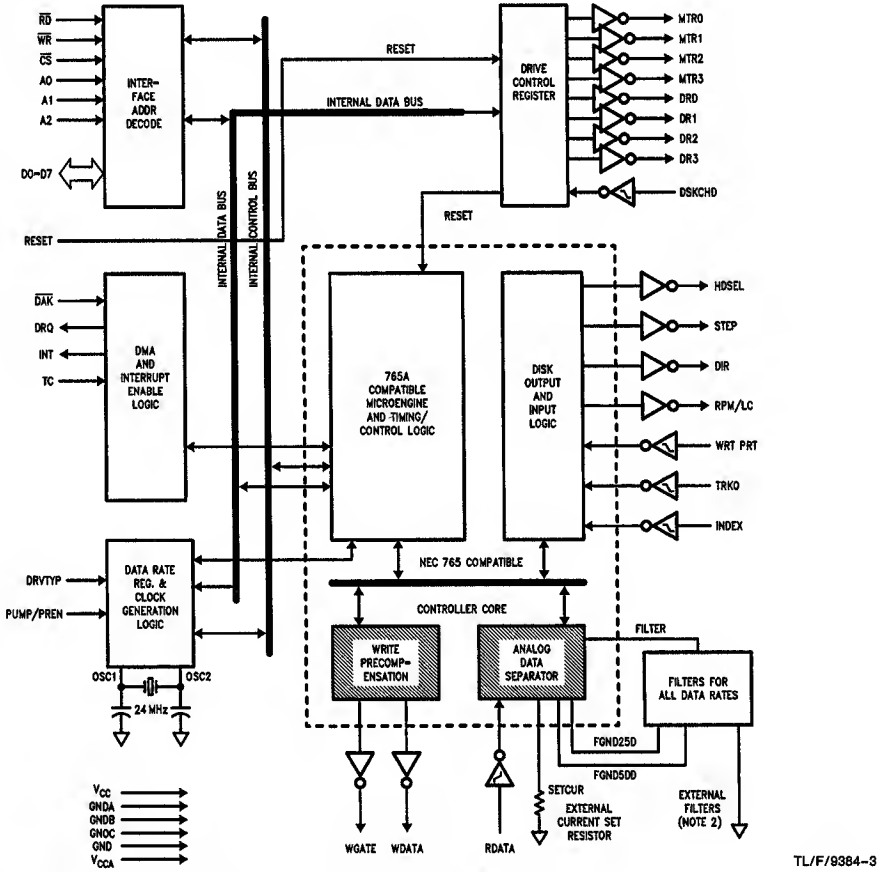
Specifically to support the PC-AT and PC-XT design, the Floppy Disk Controller PLUS-2 includes address decode for the A0–A2 address lines, the motor/drive select register, data rate register for selecting 250/300/500 kb/s, Disk Changed status, dual speed spindle motor control, low write current and DMA/interrupt sharing logic. The controller also supports direct connection to the μP bus via internal 12 mA buffers. The controller also can be connected directly to the disk drive via internal open drain high drive outputs, and Schmitt inputs.

In addition to this logic the DP8473 includes many features to ease design of higher performance drives and future controller upgrades. These include 1.0 Mb/s data rate, extended track range to 4096, Implied seeking, working Scan Commands, motor control timing, both standard IBM formats as well as Sony 3.5" (ISO) formats, and other enhancements.

This device is available in a 52 pin Plastic Chip Carrier, and in a 48 pin Dual-In-Line package.

Table of Contents

Block Diagram



TL/F/9384–3

Note 1: The MTR2, MTR3, DR2, and DR3 are not available on the 48 pin DIP (DP8473N, J) versions.

Note 2: See *Figure 4* for filter description.

Note 3: Total transistor count is 29,700 (approx).

FIGURE 1. DP8473 Functional Block Diagram

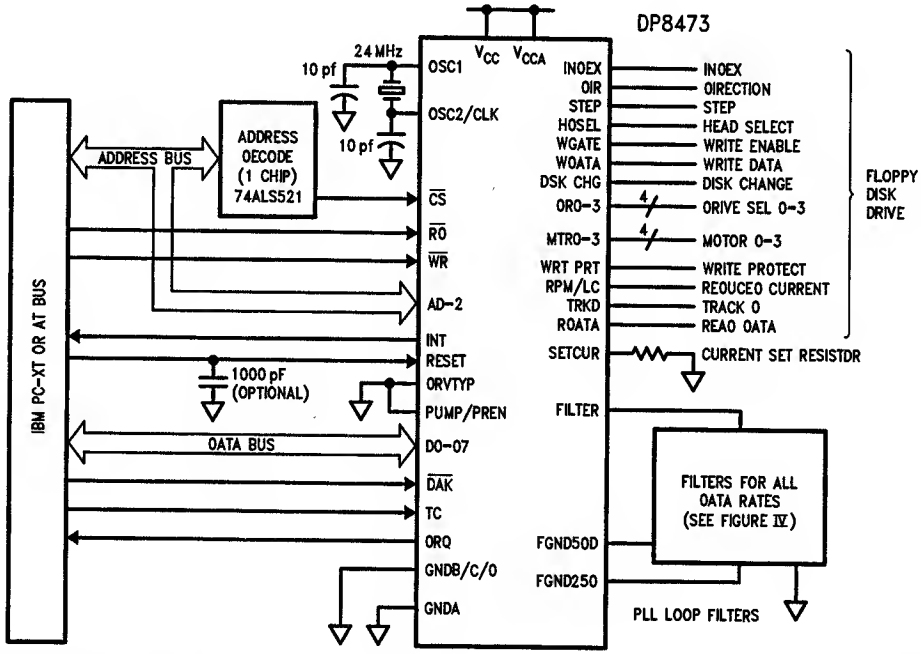
Pin Descriptions

Symbol	DP8473 PCC	DP8473 DIP	Function
MTR2	1	—	This is an active low motor enable line for drive 2, which is controlled by the Drive Control register. This is a high drive open drain output.
GNDD	2	48	This pin is the digital ground for the disk interface output drivers.
WDATA	3	1	This is the active low open drain write precompensated serial data to be written onto the selected disk drive. This is a high drive open drain output.
DIR	4	2	This output determines the direction of the head movement (low = step in, high = step out). When in the write or read modes, this output will be high. This is a high drive open drain output.
DR1	5	3	This is an active low drive select line for drive 1 that is controlled by the Drive Control register bits D0, D1. The Drive Select bit is ANDed with the Motor Enable of the same number. This is a high drive open drain output.
DR0	6	4	This is an active low drive select similar to DR1 line except for drive 0.
MTR1	7	5	This is an active low motor enable line for drive 1. Similar to MTR2.
MTR0	8	6	This is an active low motor enable line for drive 0. Similar to MTR2.
HD SEL	9	7	This output determines which disk drive head is active. Low = Head 1, Open (high) = Head 0. This is a high drive open drain output.
TRK0	10	8	This active low Schmitt input tells the controller that the head is at track zero of the selected disk drive.
INDEX	11	9	This active low Schmitt input signals the beginning of a track.
WRT PRT	12	10	This active low Schmitt input indicates that the disk is write protected. Any command that writes to that disk drive is inhibited when a disk is write protected.
V_{CCA}	13	11	This pin is the 5V supply for the analog data separator circuitry.
V_{CC}	14	12	This pin is the 5V supply for the digital circuitry.
RESET	15	13	Active high input that resets the controller to the idle state, and resets all the output lines to the disk drive to their disabled state. The Drive Control register is reset to 00. The Data Rate register is set to 250 kb/s. The Specify command registers are not affected. The Mode Command registers are set to the default values. Reset should be held active during power up. To prevent glitches activating the rest sequence, a small capacitor (1000 pF) should be attached to this pin.
$\overline{WR}$	16	14	Active low input to signal a write from the microprocessor to the controller.
$\overline{RD}$	17	15	Active low input to signal a read from the controller to the microprocessor.
$\overline{CS}$	18	16	Active low input to enable the $\overline{RD}$ and $\overline{WR}$ inputs. Not required during DMA transfers. This should be held high during DMA transfers.
A0, A1, A2	19–21	17–19	Address lines from the microprocessor. This determines which registers the microprocessor is accessing as shown in Table IV in the Register Description Section. Don't care during DMA transfers.
D0–D4	22-26	20–24	Bi-directional data lines to the microprocessor. These are the lower 5 bits and have buffered 12 mA outputs.
GNDB	27	25	This pin is the digital ground for the 12 mA microprocessor interface buffers. This includes D0–D7, INT, and DRQ.
D5–D7	28–30	26–28	Bi-directional data lines to the microprocessor. These upper 3 bits have buffered 12 mA outputs.
DRQ	31	29	Active high output to signal the DMA controller that a data transfer is needed. This signal is enabled when D3 of the Drive Control Register is set.
$\overline{DAK}$	32	30	Active low input to acknowledge the DMA request and enable the $\overline{RD}$ and $\overline{WR}$ inputs. This signal is enabled when D3 of the Drive Control Register is set.
TC	33	31	Active high input to indicate the termination of a DMA transfer. This signal is enabled when the DMA Acknowledge pin is active.
INT	34	32	Active high output to signal that an operation requires the attention of the microprocessor. The action required depends on the current function of the controller. This signal is enabled when D3 of the Drive Control Register is set.

Pin Descriptions (Continued)

Symbol	DP8473 PCC	DP8473 DIP	Function
DSKCHG/RG	35	33	This latched Schmitt input signal is inverted and routed to D7 of the data bus and is read when address xx7H is enabled. When the RG bit in the Mode Command is set, this pin functions as a Read Gate signal that when low forces the data separator to lock to the crystal, and when high it locks to data for diagnostic purposes.
GNDC	36	34	This pin is the digital ground for the controller's digital logic, including all internal registers, micro-engine, etc.
OSC2/CLOCK	37	35	One side of the external 24 MHz crystal is attached here. If a crystal is not used, a TTL or CMOS compatible clock is connected to this pin.
OSC1	38	36	One side of an external 24 MHz crystal is attached here. This pin is tied low if an external clock is used.
GNDA	39	37	This pin is the analog ground for the data separator, including all the PLLs, and delay lines.
FILTER	40	38	This pin is the output of the charge pump and the input to the VCO. One or more filters are attached between this pin and the GNDA, FGND250 and FGND500 pins.
FGND500	41	39	This pin connects the PLL filter for 500k(MFM)/250k(FM) b/s to ground. This is a low impedance open drain output.
FGND250	42	40	This pin connects the PLL filter for 250k(MFM)/125k(FM) b/s or 300k(MFM)/150k(FM) b/s to ground. This is a low impedance open drain output.
DR3	43	—	This is the same as DR0 except for drive 3.
RDATA	44	41	The active low raw data read from the disk is connected here. This is a Schmitt input.
DR2	45	—	This is the same as DR0 except for drive 2.
PUMP/PREN	46	42	When the PU bit is set in Mode Command this pin is an output that indicates when the charge pump is making a correction. Otherwise this pin is an input that sets the precomp mode as shown in Table VI. If pin is configured as PUMP, PREN is assumed high.
DRV TYP	47	43	This is an input used by the controller to enable the 300 kb/s mode. This enables the use of floppy drives with either dual or single speed spindle motors. For dual speed spindle motors, this pin is tied low. When low, and 300 kb/s data rate is selected in the data rate register, the PLL actually uses 250 kb/s. This pin is tied high for single speed spindle motor drives (standard AT drive). When this pin is high and 300 kb/s is selected 300 kb/s is used. (See also RPM/LC pin).
SETCUR	48	44	An external resistor connected from this pin to analog ground programs the amount of charge pump current that drives the external filters. The PLL Filter Design section shows how to determine the values.
WGATE	49	45	This active low open drain high drive output enables the write circuitry of the selected disk drive. This output has been designed to prevent glitches during power up and power down. This prevents writing to the disk when power is cycled.
STEP	50	46	This active low open drain high drive output will produce a pulse at a software programmable rate to move the head during a seek operation.
RPM/LC	51	47	This high drive open drain output pin has two functions based on the selection of the DRVTYP pin. 1. When using a dual speed spindle motor floppy drive (DRVTYP pin low), this output is used to select the spindle motor speed, either 300 RPM or 360 RPM. In this mode this output goes low when 250/300 kb/data rate is chosen in the data rate register, and high when 500 kb/s is chosen. 2. When using a single speed spindle motor floppy drive (DRVTYP pin high), this pin indicates when to reduce the write current to the drive. This output is high for high density media (when 500 kb/s is chosen).
MTR3	52	—	This is an active low motor enable line for drive 3.

Typical Application



Recommended Plastic Chip Carrier Socket:
AMP P/N 8215S1-1 or equivelent.

FIGURE 2. DP8473 Typical Application

Functional Description

This section describes the basic architectural features of the DP8473, and many of the enhancements provided. Refer to *Figure 1.*

765A COMPATIBLE MICRO-ENGINE

The core of the DP8473 is a μPD765A compatible microcoded engine. This engine consists of a sequencer, program ROM, and disk/misc registers. This core is clocked by either a 4 MHz, 4.8 MHz or 8 MHz clock selected in the Data Rate Register. Upon this core is added all the glue logic used to implement a PC-XT or AT, or PS/2 floppy controller, as well as the data separator and write precompensation logic.

The controller consists of a microcoded engine that controls the entire operation of the chip including coordination of data transfer with the CPU, controlling the drive controls, and actually performing the algorithms associated with reading and writing data to/from the disk. This includes the read algorithm for the data separator.

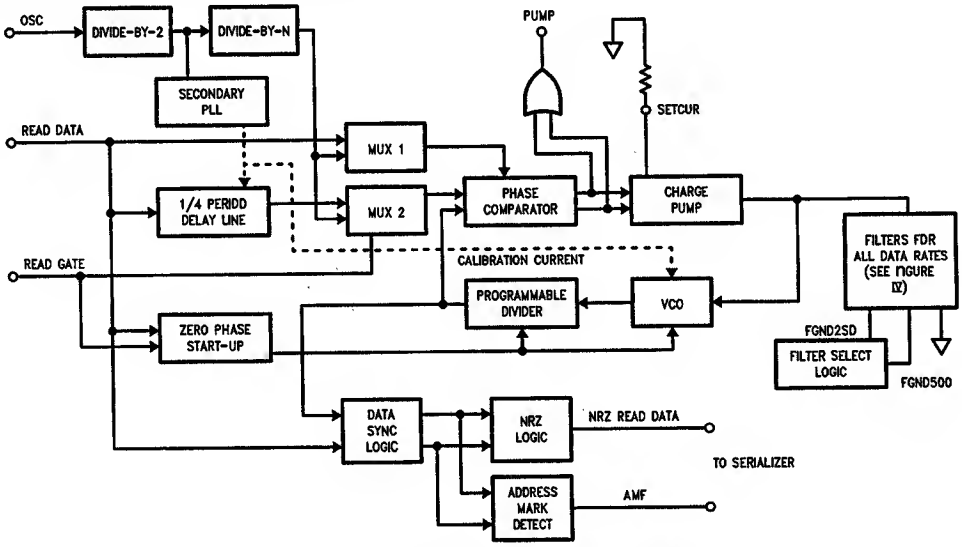
Like the μPD785A, this controller takes commands and returns data and status through the Data Register in a byte serial fashion. Handshake for command/status I/O is provided via the Main Status Register. All of the μPD765A commands are supported, as are many other enhanced commands.

DATA SEPARATOR

The internal data separator consists of an analog PLL and its associated circuitry. The PLL synchronizes the raw data signal read from the disk drive. The synchronized signal is used to separate the encoded clock and data pulses. The data pulses are de-serialized into bytes and then sent to the μP by the controller.

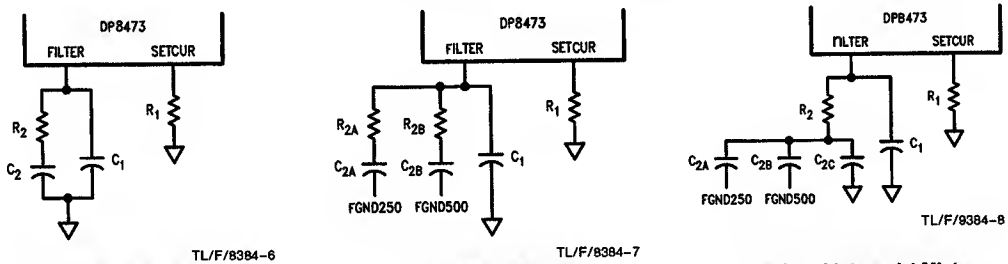
The main PLL consists of four main components, a phase comparator, a filter, a voltage controlled oscillator (VCO), and a programmable divider. The phase comparator detects the difference between the phase of the divider's output and the phase of the raw data being read from the disk. This phase difference is converted to a current which either charges or discharges one of the three external filters. The resulting voltage on the filter changes the frequency of the VCO and the divider output to reduce the phase difference between the input data and the divider's output. The PLL is "locked" when the frequency of the divider is exactly the same as the average frequency of the data read from the disk. A block diagram of the data separator is shown in *Figure 3.*

Functional Description (Continued)



TL/F/9384–5

FIGURE 3. Block Diagram of DP8473's Data Separator



TL/F/8384–6

a) Single Data Rate

TL/F/8384–7

b) 250/500 kb/s Filter

TL/F/9384–8

c) 250/500 kb/s and 1 Mb/s

Note: For all filter configurations, 250kb/s and 300 kb/s share the same filter.

FIGURE 4. Typical Configuration for Loop Filters for the DP8473 Showing Component Labels

To ensure optimal performance, the data separator incorporates several additional circuits. The quarter period delay line is used to determine the center of each bit cell. A secondary PLL is used to automatically calibrate the quarter period delay line. The secondary PLL also calibrates the center frequency of the VCO.

To eliminate the logic associated with controlling multiple data rates the DP8473 supports the connection of three filters to the chip via the FGND250, and FGND500 pins (filter ground switches). The controller chooses which filter components to use based on the value loaded in the Data Rate Register. If 500 k(MFM) is being used then the FGND500 is enabled (FGND250 is disabled). If 250 k(MFM) or 300 k(MFM) is being used the FGND250 pin is enabled, and FGND500 is disabled. For 1 Mb/s (MFM) both FGND pins are disabled.

Note for FM encoding: Sometimes, after a reset, the DP8473 will consistently return an error in the Result Phase after an FM read command. If this occurs, simply reset the DP8473 and retry the operation. This may have to be done more than once, as many as five times. Resetting and repeating will prevent soft errors being reported prematurely. This technique is used by MS-DOS.

Figure 4 shows several possible filter configurations. For a filter to cover all data rates (*Figure 4c*), the DP8473 has a 1 Mb/s filter always connected and other capacitor filter components for the other data rates are switched in parallel to this filter. The actual loop filter for 500 kb/s is the parallel combination of the two capacitors, C_{2C} and C_{2B}, attached to the FGND500 pin and to ground. The 250/300 kb/s filter is the parallel combination of the capacitors, C_{2C} and C_{2A}, attached to the FGND250, and ground. If 1 Mb/s need not be supported then the filter configuration of *Figure 4b* can be used. This configuration allows more optimal performance for both 500k and 250/300 kb/s. *Figure 4a* is a simple filter configuration primarily for a single data rate (or multiple data rates with a performance compromise). Table ii shows some typical filter values. Other filter configurations and values are possible, these result in good general performance.

While the controller and data separator support both FM and MFM encoding, the filter switch circuitry only supports

Functional Description (Continued)

the IBM standard MFM data rates. To provide both FM and MFM filters external logic may be necessary.

The controller takes best advantage of the internal data separator by implementing a sophisticated ID search algorithm. This algorithm, shown in *Figure 5*, enhances the PLL's lock characteristics by forcing the PLL to relock to the crystal any time the data separator attempts to lock to a non-preamble pattern. This algorithm ensures that the PLL is not thrown way out of lock by write splices or bad data fields.

TABLE II. Typical Filter Values for the Various Data Rates (Assuming ±6% Capture Range)

Data Rate (MFM b/s)	C_2	R_2	C_1	R_1
Filter Values when Using All 3 Data Rates				
1.0M 500k 250/300k	C_{2C} = 0.012 μF C_{2B} = 0.015 μF C_{2A} = 0.033 μF	560Ω	510 pF	5.6 kΩ
Filter Values when Using 250/300 and 500 kb/s				
500k 250/300k	C_{2B} = 0.027 μF C_{2A} = 0.047 μF	560Ω 560Ω	1000 pF	5.6 kΩ
Filter Using Only One Data Rate				
1.0M 500k 300/250k	C_2 = 0.012 μF C_2 = 0.027 μF C_2 = 0.047 μF	560Ω 560Ω 560Ω	510 pF 1000 pF 2000 pF	5.6 kΩ 5.6 kΩ 5.6 kΩ

(These values are preliminary and thus are subject to change.)

TABLE III. Data Rates (MFM) versus VCO Divide-By Factor

Data Rate	N
1 Mb/s	4
500 kb/s	8
300 kb/s	16
250 kb/s	16

PLL DIAGNOSTIC MODES

In addition, the DP8473 has two diagnostic modes to enable filter optimization, 1) enabling the Charge Pump output signal onto the PUMP/PREN pin, and 2) providing external control of the Read Gate signal to the data separator. Both modes are enabled in the last byte of the Mode Command.

The Pump output signal indicates when the charge pump is making a phase correction, and hence whether the loop is locked or not.

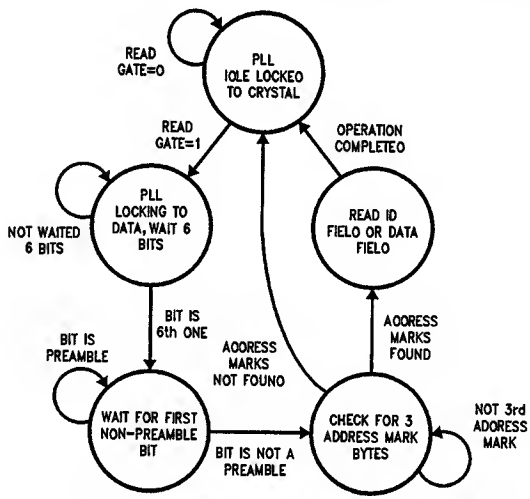
The Read Gate function, when enabled, allows the designer to manually force the data separator to lock to the incoming data or back to the reference clock. This enables easy verification of the lock characteristics of the PLL, by monitoring the FILTER pin, and the Pump signal.

PLL FILTER DESIGN

This section provides information to enable design of the data separator's external filter and charge pump set resistor. This discussion is for a single data rate filter, and can be easily extrapolated to the other filters of *Figure 4*. Table II shows some typical filter component values, but if a custom filter is desired, the following parameters must be considered:

R_1: Charge pump current setting resistor. The current set by this resistor is multiplied by the charge pump gain, K_P which is ~2.5. Thus the charge pump current is:

I_{PUMP} = (2.5) 1.2V/R_1. R_1 should be set to between 3–12 kΩ. This resistor determines the gain of the phase detector, which is $K_D = I_{PUMP}/2\pi$.



TL/F/9384–9

FIGURE 5. Read Algorithm-State Diagram for Data

Functional Description (Continued)

C_2: Filter capacitor in series with R_2. With pump current this determines loop bandwidth.

R_2: Filter resistor. Determines the PLL damping factor.

C_1: This filter capacitor improves the performance of the PLL by providing additional filtering of bit jitter and noise.

K_{VCO}: The ratio of the change in the frequency of the VCO output due to a voltage change at the VCO input. $K_{VCO} \approx 25$ Mrad/s/V. The VCO is followed by a divider to achieve the desired frequency for each data rate. VCO center frequency is 4 MHz for data rates of 1 Mb/s, 500 kb/s, and 250 kb/s (MFM), and is 4.8 MHz for 300 kb/s (MFM).

K_{PLL}: This is the gain of the internal PLL circuitry, and is the product of $V_{REF} \times K_{VCO} \times K_P$. This value is specified in the Phase Locked Loop Characteristics table.

ω_n: This is the bandwidth of the PLL, and is given by,

$$\omega_n = \sqrt{\frac{K_{PLL}}{2\pi C_2 N R_1}}$$

where N is the number of VCO cycles between two phase comparisons. The value of N for the various data rates are shown in Table III.

ζ: The damping factor is set to 0.7 to 1.2 and is given by,

$$\zeta = \frac{\omega_n R_2 C_2}{2}$$

The trade off, when choosing filter components is between acquisition time while the PLL is locking and jitter immunity while reading data. To select the proper components for a standard floppy disk application the following procedure can be used:

1. Choose FM or MFM, and data rate. Determine N from Table III. Determine preamble length (MFM = 12). The PLL should lock within ½ the preamble time.

2. Determine loop bandwidth (ω_n) required, and set the charge pump resistor R_1.

3. Calculate C_2 using:

$$C_2 = \frac{K_{PLL}}{2\pi R_1 N \omega_n^2}$$

4. Choose R_2 using:

$$R_2 = \frac{2\zeta}{\omega_n C_2}$$

6. Select C_1 to be about $\frac{1}{20}$th of C_2.

The above procedure will yield adequate loop performance. If optimum loop performance is required, or if the nature of the loop performance is very critical, then some additional consideration must be given to choosing ω_n and the damping factor. (For a detailed description on how to choose ω_n and ζ, see: **AN-505 Floppy Disk Data Separator Design Guide for the DP8473).**

WRITE PRECOMPENSATION

The DP8473 incorporates a single fixed 3-bit shift register. This shift register outputs are tapped and multiplexed onto the write data output. The taps are selected by a standard precompensation algorithm. This precompensation value can be selected from the PUMP/PREN pin. When this pin is low 125 ns precomp is used for all data rates except 1 Mb/s which uses 83 ns. When PREN is tied high, the precompensation-value scales with data rate at 250 kb/s its 250 ns, for 300 kb/s its 208 ns, at 500 kb/s its 125 ns, and at 1.0 Mb/s its 83 ns. These values are shown in Table VI.

PC-AT AND PC-XT LOGIC BLOCKS

This section describes the major functional blocks of the PC logic that have been integrated on the controller. Refer back to *Figure 1,* the block diagram.

DMA Enable Logic: This is gating logic that disables the DMA lines and the Interrupt output, under the control of the DMA Enable bit in the Drive control register. When the DMA Enable bit is 0 then the INT, and DRQ are held TRI-STATE, and $\overline{DAK}$ is disabled.

Drive Output Buffers/Input Receivers: The drive interface output pins can drive 150Ω ±10% termination resistors. This enables connection to a standard floppy drive. All drive interface inputs are TTL compatible schmitt trigger inputs with typically 250 mV of hysteresis. *The only functional differences between the 52 pin PLCC and the 48 pin DIP version are that the MTR2 and 3, and DR2 and 3 pins have been removed in order to accommodate the 48 pin package.*

Bus Interface-Address Decode: The address decode circuit allows software access to the controller, Drive Control Register, and Data Rate Register (see Table IV for the memory map) using the same address map as is used in the XT, AT, or PS/2. The decoding is provided for A0–A2, so only a single address decoder connected to the chip select is needed to complete the decode. The bus interface logic includes the 8-bit data bus and DRQ/INT signals. The output drive for these pins is 12 mA.

TABLE IV. Address Memory Map for DP8473

A2	A1	A0	R/W	Register
0	0	0	X	None (Bus TRI-STATE)
0	0	1	X	None (Bus TRI-STATE)
0	1	0	W	Drive Control Register
0	1	1	X	None (Bus TRI-STATE)
1	0	0	R	Main Status Register
1	0	1	R/W	Data Register
1	1	0	X	None (Bus TRI-STATE)
1	1	1	W	Data Rate Register
1	1	1	R	Disk Changed Bit*

*When this location is accessed only bit D7 is driving, all others are held TRI-STATE.

Drive Control Register: This 8-bit write only register controls the drive selects, motor enables, DMA enable, and Reset. See Register Description.

Reset Logic: The reset input pin is active high, and directly feeds the Drive Control Register and the Data Rate Register. After a hardware reset the Drive Control Register is reset to all zeros, and the Data Rate Register is set to 250 kb/s data rate. The controller is held reset until the software sets the Drive Control reset bit, after which the controller may be initialized. A software reset to the controller core can be issued by resetting then setting this bit. A software reset does not reset the Drive Control Register, or the Data Rate Register.

Functional Description (Continued)

Data Rate Register and Clock Logic: This is a two bit register that controls the data rate that the controller uses. See Register Description. This register feeds logic that selects the data rates by programming a prescaler that divides the crystal or clock input by either 3, 5, or 6. This causes either 4 MHz, 4.8 MHz and 8 MHz to be input as the master clock for the controller core. If the Drive Type pin is high and a 300 kb/s data rate is chosen, 4.8 MHz is used to generate 300 kb/s, but when the DRVTYP pin is low and 300 kb/s is selected, 4 MHz is used, and the actual data rate is 250 kb/s. See Table VI.

Low Power Mode Logic: This logic is an enhancement over the standard XT, AT, PS/2 design. In the Low Power Mode the crystal oscillator, controller and all linear circuitry are turned off. When the oscillator is turned off the controller will typically draw about 100 μA. The internal circuitry is disabled while the oscillator is off because the internal circuitry is driven from this clock. The oscillator will turn back on automatically after it detects a read or a write to the Main Status or Data Registers. It may take a few milli-seconds for the oscillator to stabilize and the μP will be prevented from trying to access the Data Register during this time through the normal Main Status Register protocol. (The Request for Master bit in the Main Status Register will be inactive.) There are two ways to go into the low power mode. One is to command the controller to switch to low power immediately. The other method is to set the controller to automatically go into the low power mode 500 ms after the beginning of the idle state (based on a 500 kb/s (MFM) data rate). This would be invisible to the software. The low power mode is programmed through the Mode Command.

The Data Rate Register and the Drive Control Register are unaffected by the power down mode. They will remain active. It is up to the user to ensure that the Motor and Drive select signal are turned off.

TABLE V. Truth Table for Drive Control Register

D7	D6	D5	D4	D1	D0	Function
X	X	X	1	0	0	Drive 0 Selected (DR0 = 0)
X	X	1	X	0	1	Drive 1 Selected (DR1 = 0)
X	1	X	X	1	0	Drive 2 Selected (DR2 = 0)
1	X	X	X	1	1	Drive 3 Selected (DR3 = 0)

Crystal Oscillator: The DP8473 is clocked by a single 24 MHz signal. An on-chip oscillator is provided, to enable the attachment of a crystal, or a clock. If a crystal is used, a 24 MHz fundamental mode, parallel resonant crystal should be used. This crystal should be specified to have less than 150Ω series resistance, and shunt capacitance of less than 7 pF. Typically a series resonant crystal can be used, it will just oscillate in parallel mode 30–300 ppm from its ideal frequency.

If an external oscillator circuit is used, it must have a duty cycle of at least 40–60%, and minimum input levels of 2.4V and 0.4V. The controller should be configured so that the clock is input into the OSC2 pin, and OSC1 is tied to ground.

Crystals: Staytek: CX1-SM1-24 MHz(B)
SaRonix: SRX 3164

Register Description

This section describes the register bits for all the registers that are directly accessible to the μP. Table IV (previous page) shows the memory map for these registers. Note that in the PC some of the registers are partially decoded, this is not the case here. All registers occupy only their documented addresses.

MAIN STATUS REGISTER (Read Only)

The read only Main Status Register indicates the current status of the disk controller. The Main Status Register is always available to be read. One of its functions is to control the flow of data to and from the Data Register. The Main Status Register indicates when the disk contoller is ready to send or receive data. It should be read before each byte is transferred to or from the Data Register except during a DMA transfer. No delay is required when reading this register after a data transfer.

D7 Request for Master: Indicates that the Data Register is ready to send or receive data from the μP. This bit is cleared immediately after a byte transfer and will become set again as soon as the disk controller is ready for the next byte.

D6 Data Direction: Indicates whether the controller is expecting a byte to be written to (0) or read from (1) the Data Register.

D5 Non-DMA Execution: Bit is set only during the Execution Phase of a command if it is in the non-DMA mode. In other words, if this bit is set, the multiple byte data transfer (in the Execution Phase) must be monitored by the μP either through interrupts, or software polling as described in the Processor Software Interface section.

D4 Command In Progress: Bit is set after the first byte of the Command Phase is written. Bit is cleared after the last byte of the Result Phase is read. If there is no result phase in a command, the bit is cleared after the last byte of the Command Phase is written.

D3 Drive 3 Seeking: Set after the last byte of the Command Phase of a Seek or Recalibrate command is issued for drive 3. Cleared after reading the first byte in the Result Phase of the Sense Interrupt Command for this drive.

D2 Drive 2 Seeking: Same as above for drive 2.

D1 Drive 1 Seeking: Same as above for drive 1.

D0 Drive 0 Seeking: Same as above for drive 0.

DATA REGISTER (Read/Write)

This is the location through which all commands, data and status flow between the CPU and the DP8473. During the Command Phase the μP loads the controller's commands into this register based on the Status Register Request for Master and Data Direction bits. The Result Phase transfers the Status Registers and header information to the μP in the same fashion.

Register Description (Continued)

TABLE VI. Deta Rata and Precompeneation Programming Velues

D1	D0**	DRVTYP Pin	Deta Rate MFM (kb/s)	Normal Precomp* (ns)	Alternate Pracomp* (ne)	FGND Pin Enebled	RPM/LC Pin Level
0	0	X	500	125	125	FGND500	High
0	1	0	250	125	250	FGND250	Low
0	1	1	300	208	208	FGND250	Low
1	0	0	250	125	250	FGND250	Low
1	0	1	250	125	250	FGND250	Low
1	1	0	1000	83	83	None	High
1	1	1	1000	83	83	None	Low

*Normal values when PUMP/PREN pin set low; Altarnata values when PUMP/PREN pin set high.

**D0 and D1 ara Data Rste Control Bits.

DRIVE CONTROL REGISTER (Write Only)

D7 Motor Enabla 3: This controls the Motor for drive 3, MTR3. When 0 the output is high, when 1 the output is low. (Note this signal is not output to a pin on 48 pin DIP version.)

D6 Motor Enable 2: Same function as D7 except for drive 2's motor. (Note this signal is not brought out to a pin on DIP.)

D5 Motor Enable 1: This bit controls the Motor for drive 1's motor. When this bit is 0 the MTR1 output is high.

D4 Motor Enable 0: Same as D5 except for drive 0's motor.

D3 DMA Enabla: When set to a 1 this enables the DRQ, DAK, INT pins. A zero disables these signals.

D2 Rasat Controller: This bit when set to a 0 resets the controller, and when a 1 enables normal operation. It does not affect the Drive Control or Data Rate Registers which are reset only by a hardware reset.

D1–D0 Drive Salect: These two pins are encoded for the four drive selects, and are gated with the motor enable lines, so that only one drive is selected when it's Motor Enable is active. (See Table V.)

DATA RATE REGISTER (Write Only)

D7–D2: Not used.

D1, D0 Date Rate Salect: These bits set the data rate and the write precompensation values for the disk controller. After a hardware reset these bits are set to 10 (250 kb/s). They are encoded as shown in Table VI.

DISK CHANGED REGISTER (Read Only)

D7 Disk Changed: This bit is the latched complement of the Disk Changed input pin. If the DSKCHG input is low this bit is high.

D6–D0: These bits are reserved for use by the hard disk controller, thus during a read of this register, these bits are TRI-STATE.

Result Phase Status Registers

The Result Phase of a command contains bytes that hold status information. The format of these bytes are described below. Do not confuse these register bytes with the Main Status Register which is a read only register that is always available. The Result Phase status registers are read from the Data Register only during the Result Phase.

STATUS REGISTER 0 (ST0)

D7–D6 Intarrupt Coda:

00 = Normal Termination of Command.

01 = Abnormal Termination of Command. Execution of Command was started, but was not successfully completed.

10 = Invalid Command Issue. Command Issued was not recognized as a valid command.

11 = Ready changed state during the polling mode.

D5 Saak End: Seek or Recalibrate Command completed by the Controlier. (Used during Sense Interrupt command.)

D4 Equipmant Check: After a Recalibrate Command, Track 0 signal failed to occur. (Used during Sense Interrupt command.)

D3 Not Usad: 0

D2 Haad Addrass (at end of Execution Phase).

D1, D0 Driva Salect (at end of Execution Phase).

00 = Drive 0 selected. 01 = Drive 1 selected.

10 = Drive 2 selected. 11 = Drive 3 selected.

STATUS REGISTER 1 (ST1)

D7 End of Track: Controller transferred the last byte of the last sector without the TC pin becoming active. The last sector is the End Of Track sector number programmed in the Command Phase.

D6 Not Used: 0

D5 CRC Error: If this bit is set and bit 5 of ST2 is clear, then there was a CRC error in the Address Field of the correct sector. If bit 5 of ST2 is set, then there was a CRC error in the Data Field.

D4 Over Run: Controller was not serviced by the μP soon enough during a data transfer in the Execution Phase.

Result Phase Status Registers (Continued)

TABLE VII. Maximum Time Allowed to Service an Interrupt or Acknowledge a DMA Request In Execution Phase

Data Rate	Time to Service
125	62.0 μs
250	30.0 μs
500	14.0 μs
1000	6.0 μs

Time from rising edge of DRQ or INT to trailing edge of $\overline{DAK}$ or $\overline{RD}$ or $\overline{WR}$.

D3 Not Used: 0

D2 No Data: Three possible problems: 1) Controller cannot find the sector specified in the Command Phase during the execution of a Read, Write, or Scan command. An address mark was found however so it is not a blank disk. 2) Controller cannot read any Address Fields without a CRC error during Read ID command. 3) Controller cannot find starting sector during execution of Read A Track command.

D1 Not Writable: Write Protect pin is active when a Write or Format command is issued.

D0 Missing Address Mark: If bit 0 of ST2 is clear then the disk controller cannot detect any Address Field Address Mark after two disk revolutions. If bit 0 of ST2 is set then the disk controller cannot detect the Data Field Address Mark.

STATUS REGISTER 2 (ST2)

D7 Not Used: 0

D6 Control Mark: Controller tried to read a sector which contained a deleted data address mark during execution of Read Data or Scan commands. Or, if a Read Deleted Data command was executed, a regular address mark was detected.

D5 CRC Error in Data Field: Controller detected a CRC error in the Data Field. Bit 5 of ST1 is also set.

D4 Wrong Track: Only set if desired sector not found, and the track number recorded on any sector of the current track is different from that stored in the Track Register.

D3 Scan Equal Hit: "Equal" condition satisfied during any Scan Command.

D2 Scan Not Satisfied: Controller cannot find a sector on the track which meets the desired condition during Scan Command.

D1 Bad Track: Only set if the desired sector is not found, and the track number recorded on any sector on the track is different from that stored in the Track Register and the recorded track number is FF.

D0 Missing Address Mark in Data Field: Controller cannot find the Data Field Address Mark during Read/Scan command. Bit 0 of ST1 is also set.

STATUS REGISTER 3 (ST3)

D7 Not Used: 0

D6 Write Protect Status

D5 Not Used: 1

D4 Track 0 Status

D3 Not Used: 0

D2 Head Select Status

D1, D0 Drive Selected:

00 = Drive 0 selected. 01 = Drive 1 selected.

10 = Drive 2 selected. 11 = Drive 3 selected.

Processor Software Interface

Bytes are transferred to and from the disk controller in different ways for the different phases in a command.

COMMAND SEQUENCE

The disk controller can perform various disk transfer, and head movement commands. Most commands involve three separate phases.

Command Phase: The μP writes a series of bytes to the Data Register. These bytes indicate the command desired and the particular parameters required for the command. All the bytes must be written in the order specified in the Command Description Table. The Execution Phase starts immediately after the last byte in the Command Phase is written. Prior to performing the Command Phase, the Drive Control and Data Rate Registers should be set.

Execution Phase: The disk controller performs the desired command. Some commands require the μP to read or write data to or from the Data Register during this time. Reading data from a disk is an example of this.

Result Phase: The μP reads a series of bytes from the data register. These bytes indicate whether the command executed properly and other pertinent information. The bytes are read in the order specified in the Command Description Table.

A new command may be initiated by writing the Command Phase bytes after the last bytes required from the Result Phase have been read. If the next command requires selecting a different drive or changing the data rate the Drive Control and Data Rate Registers should be updated. If the command is the last command, then the software should deselect the drive. *(Note as a general rule the operation of the controller core is independent of how the μP updates the Drive Control and Data Rate Registers. The software must ensure that manipulation of these registers is coordinated with the controller operation.)*

During the Command Phase and the Result Phase, bytes are transferred to and from the Data Register. The Main Status Register is monitored by the software to determine when a data transfer can take place. Bit 6 of the Main Status Register must be clear and bit 7 must be set before a byte can be written to the Data Register during the Command Phase. Bits 6 and 7 of the Main Status Register must both be set before a byte can be read from the Data Register during the Result Phase.

If there is information to be transferred during the Execution Phase, there are three methods that can be used. The DMA mode is used if the system has a DMA controller. This allows the μP to do other things during the Execution Phase data transfer. If DMA is not used, an interrupt can be issued for each byte transferred during the Execution Phase. If interrupts are not used, the Main Status Register can be polled to indicate when a byte transfer is required.

Processor Software Interface (Continued)

DMA MODE

If the DMA mode is selected, a DMA request will be generated in the Execution Phase when each byte is ready to be transferred. To enable DMA operations during the Execution Phase, the DMA mode bit in the Specify Command must be enabled, and the DMA signals must be enabled in the Drive Control Register. The DMA controller should respond to the DMA request with a DMA acknowledge and a read or write strobe. The DMA request will be cleared by the active edge of the DMA acknowledge. After the last byte is transferred, an interrupt is generated, indicating the beginning of the Result Phase. During DMA operations the Chip Select input must be held high. TC is asserted to terminate an operation. Due to the internal gating TC is only recognized when the $\overline{DAK}$ input is low.

INTERRUPT MODE

If the non-DMA mode is selected, an interrupt will be generated in the Execution Phase when each byte is ready to be transferred. The Main Status Register should be read to verify that the Interrupt is for a data transfer. Bits 5 and 7 of the Main Status Register will be set. The interrupt will be cleared when the byte is transferred to or from the Data Register. The μP should transfer the byte within the time allotted by Table VII. If the byte is not transferred within the time allotted, an Overrun Error will be indicated in the Result Phase when the command terminates at the end of the current sector.

An interrupt will also be generated after the last byte is transferred. This indicates the beginning of the Result Phase. Bits 7 and 6 of the Main Status Register will be set and bit 5 will be clear. This interrupt will be cleared by reading the first byte in the Result Phase.

SOFTWARE POLLING

If the non-DMA mode is selected and interrupts are not suitable, the μP can poll the Main Status Register during the Execution Phase to determine when a byte is ready to be transferred. In the non-DMA mode, bit 7 of the Main Status Register reflects the state of the interrupt pin. Otherwise, the data transfer is similar to the Interrupt Mode described above.

Command Description Table

READ DATA

Command Phase

MT	MFM	SK	0	0	1	1	0
IPS	X	X	X	X	HD	DR1	DR0
Track Number							
Drive Heed Number							
Sector Number							
Number of Bytes per Sector							
End of Treck Sector Number							
Intersector Gep Length							
Dete Length							

Note 1

Result Phase

Stetus Register 0
Status Register 1
Stetus Register 2
Treck Number
Head Number
Sector Number
Bytes/Sector

READ ID

Command Phase

0	MFM	0	0	1	0	1	0
X	X	X	X	X	HD	DR1	DR0

Result Phase

Status Register 0
Stetus Register 1
Stetus Register 2
Treck Number
Head Number
Sector Number
Bytes/Sector

FORMAT A TRACK

Command Phase

0	MFM	0	0	1	1	0	1
X	X	X	X	X	HD	DR1	DR0
Number of Bytes per Sector							
Number of Sectors per Track							
Intersector Gep Length							
Deta Pettern							

Result Phase

Stetus Register 0
Stetus Register 1
Status Register 2
Treck Number
Heed Number
Sector Number
Bytes/Sector

Command Description Table (Continued)

READ DELETED DATA

Command Phase

MT	MFM	SK	0	1	1	0	0
IPS	X	X	X	X	HD	DR1	DR0
Track Number							
Drive Head Number							
Sector Number							
Number of Bytes per Sector							
End of Track Sector Number							
Intersector Gap Length							
Dete Length							

Result Phase

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Bytes/Sector

WRITE DATA

Command Phase

MT	MFM	0	0	0	1	0	1
IPS	X	X	X	X	HD	DR1	DR0
Track Number							
Drive Head Number							
Sector Number							
Number of Bytes per Sector							
End of Track Sector Number							
Intersector Gap Length							
Data Length							

Result Phase

Status Register 0
Status Register 1
Stetus Register 2
Track Number
Heed Number
Sector Number
Bytes/Sector

SCAN EQUAL

Command Phase

MT	MFM	SK	1	0	0	0	1
IPS	X	X	X	X	HD	DR1	DR0
Treck Number							
Drive Head Number							
Sector Number							
Number of Bytes per Sector							
End of Track Sector Number							
Intersector Gap Length							
Sector Step Size							

Result Phase

Status Register 0
Status Register 1
Status Register 2
Track Number
Heed Number
Sector Number
Bytes/Sector

READ A TRACK

Command Phase

0	MFM	SK	0	0	0	1	0
IPS	X	X	X	X	HD	DR1	DR0
Treck Number							
Drive Head Number							
Sector Number							
Number of Bytes per Sector							
End of Track Sector Number							
Intersector Gep Length							
Data Length							

Result Phase

Status Register 0
Status Register 1
Stetus Register 2
Treck Number
Haed Number
Sector Number
Bytes/Sector

WRITE DELETED DATA

Command Phase

MT	MFM	0	0	1	0	0	1
IPS	X	X	X	X	HD	DR1	DR0
Track Number							
Drive Heed Number							
Sector Number							
Number of Bytes per Sector							
End of Treck Sector Number							
Intersector Gep Length							
Data Length							

Result Phase

Status Register 0
Status Register 1
Status Register 2
Treck Numbar
Head Number
Sector Number
Bytes/Sector

SCAN LOW OR EQUAL

Command Phase

MT	MFM	SK	1	1	0	0	1
IPS	X	X	X	X	HD	DR1	DR0
Track Number							
Drive Head Number							
Sector Number							
Number of Bytes per Sector							
End of Track Sector Number							
Intersector Gap Length							
Sector Step Size							

Result Phase

Stetus Register 0
Status Register 1
Stetus Register 2
Track Number
Head Number
Sector Number
Bytes/Sector

Command Description Table (Continued)

SCAN HIGH OR EQUAL

Command Phase

MT	MFM	SK	1	1	1	0	1
IPS	X	X	X	X	HD	DR1	DR0
Track Number							
Drive Head Number							
Sector Number							
Number of Bytes per Sector							
End of Track Sector Number							
Intersector Gap Length							
Sector Step Size							

Result Phase

Status Register 0
Status Register 1
Status Register 2
Treck Number
Head Number
Sector Number
Bytes/Sector

SEEK

Command Phase

0	0	0	0	1	1	1	1
X	X	X	X	X	X	DR1	DR0
New Track Number							
MSB of Track				0	0	0	0

Note 2

RECALIBRATE

Command Phase

0	0	0	0	0	1	1	1
0	0	0	0	0	0	DR1	DR0

SENSE INTERRUPT

Command Phase

0	0	0	0	1	0	0	0

Result Phase

Status Register 0							
Present Track Number (PTN)							
MSN PTN				0	0	0	0

Note 2

SENSE DRIVE STATUS

Command Phase

0	0	0	0	0	1	0	0
X	X	X	X	X	HD	DR1	DR0

Result Phase

Status Register 3

SPECIFY

Command Phase

0	0	0	0	0	0	1	1
Step Rate Time				Motor Off Time			
Motor On Time							DMA

MODE

Command Phase

0	0	0	0	0	0	0	1
TMR	IAF	IPS	0	LW	PR	1	ETR
0	0	0	0	0	0	0	0
1	1	0	WLD	Heed Settle			
0	0	0	0	0	RG	0	PU

Note 3

SET TRACK

Command Phase

0	$\overline{R}$/W	1	0	0	0	0	1
0	0	1	1	0	MSB	DR1	DR0
New Track Number							

Result Phase

Value

Note 3

INVALID COMMAND

Command Phase

Invalid Op Codes

Result Phase

Status Register 0

Note 1: The IPS bit is only enabled if the IPS bit in the mode command is set. Otherwise this bit is a don't care.

Note 2: Sheded byte only written or reed if the extended treck range mode is enabled in the Mode Command (ET) = 1.

Note 3: These commends ere eddltionel enhanced commends.

Note: Mnemonic Definitions

X = DON'T CARE

MFM = Dets Encoding Scheme

MSN PTN = Most Significant Nibble Present Treck Number

MT = Muiti-Treck

IPS = Implied Seek (in individual commands this bit is e don't cere unless the IPS bit in the mode commend is set.)

SK = Skip Sector

HD = Head Number

DRn = Drive to Select (encoded)

TMR = Motor/Heed Timer Mode

IAF = Index Address Field

LW PR = Low Power Mode

ETR = Extended Track Range

WLD = Wiidcerd in Scen

RG = Enebles the Read Gste input on the DSKCHG pin for the Deta Separator.

PU = Enebles Cherge Pump PUMP signei to be output on the PUMP/PREN pin.

MSB = Selects whether the most significant or least significant byte of the track is read. 1 = MSB.

$\overline{R}$/W = Selects whether the treck is written or read (Read = 0, Write = 1).

Command Description

READ DATA

The Reed Dete op-code Is written to the dete register followed by 8 bytes es specified In the Commend Description Table. After the last byte is written, the controller starts looking for the correct sector header. Once the sector is found the controller sends the dete to the μP. After one sector is finished, the Sector Number is incremented by one end this new sector is searched for. If MT (Multi-Treck) is set, both sides of one track can be read. Starting on side zero, the sectors are read until the sector number specified by End of Treck Sector Number is reeched. Then, side one Is read starting with sector number one.

In DMA mode the Reed Dete commend continues to reed until the TC pin is set. This meens that the DMA controller should be programmed to transfer the correct number of bytes. TC could be controlled by the μP and be asserted when enough bytes are received. An alternative to these methods of stopping the Read Data command is to program the End of Track Sector Number to be the last sector number that needs to be read. The controller will stop reading the disk with an error indicating that it tried to access a sector number beyond the end of the treck.

The Number of Deta Bytes per Sector parameter is defined in Teble VIII. If this is set to zero then the Data Length parameter determines the number of bytes that the controller transfers to the μP. If the data length specified is smaller than 128 the controller still reads the entire 128 byte sector end checks the CRC, though only the number of bytes specified by the Dete Length paremeter are transferred to the μP. Data Length should not be set to zero. If the Number of Bytes per Sector perameter is not zero, the Date Length peremeter hes no meening end should be set to FF (hex).

If the Implied Seek Mode is enabled by both the Mode commend end the IPS bit in this commend, e Seek will be performed to the treck number specified in the Command Phase. The controller will also wait the Head Settle time if the implied seek is enebled.

After all these conditions ere met, the controller seerches for the specified sector by comparing the track number, head number, sector number, and number bytes/sector given In the Commend Phase with the appropriate bytes read off the disk in the Address Fields.

If the correct sector is found, but there is a CRC error in the Address Field, bit 5 of ST1 (CRC Error) is set and an abnormal termination is indicated. If the correct sector is not found, bit 2 of ST1 (No Deta) is set end en ebnormal termination is indicated. In addition to this, if any Address Field track number is FF, bit 1 of ST2 (Bad Treck) is set or if any Address Field treck number is different from thet specified in the Command Phese, bit 4 of ST2 (Wrong Treck) is set.

After finding the correct sector, the controller reeds thet Deta Field. If e Deleted Dete Mark Is found end the SK bit Is set, the sector is not read, bit 6 of ST2 (Control Mark) is set, and the next sector is searched for. If a deleted data merk is found end the SK bit is not set, the sector is read, bit 6 of ST2 (Control Merk) is set, end the reed terminetes with a normal termination. If e CRC error is detected In the Data Field, bit 5 is set in both ST1 and ST2 (CRC Error) and an ebnormal terminetion is indicated.

If no problems occur in the reed commend, the reed will continue from one sector to the next in logical order (not physical order) until either TC Is set or en error occurs.

If a disk has not been inserted into the disk drive, there are many opportunities for the controller to appear to heng up. It does this if it is waiting for e certein number of disk revolutions for something. If this occurs, the controller cen be forced to ebort the command by writing a byte to the Data register. This will piece the controller into the Result Phese.

TABLE VIII. Sector Size Selection

Bytes/Sector Code	Number of Bytes In Data Field
0	128
1	256
2	512
3	1024
4	2048
5	4096
8	8192

An interrupt will be generated when the Execution Phase of the Read Deta commend terminetes. The values that will be reed back in the Result Phase are shown in Teble IX. If en error occurs, the result bytes will indicate the sector being read when the error occurred.

READ DELETED DATA

This commend is the same as the Read Data commend except for its treetment of e Deleted Date Mark. If e Deleted

TABLE IX. Result Phase Termination Values with No Error

MT	HD	Last Sector	ID Information at Result Phase			
			Track	Head	Sector	B/S
0	0	< EOT	NC	NC	S+1	NC
0	0	= EOT	T+1	NC	1	NC
0	1	< EOT	NC	NC	S+1	NC
0	1	= EOT	T+1	NC	1	NC
1	0	< EOT	NC	NC	S+1	NC
1	0	= EOT	NC	1	1	NC
1	1	< EOT	NC	NC	S+1	NC
1	1	= EOT	T+1	0	1	NC

EOT = End of Track Sector Number from Command Phase

NC = No Change in Value

S = Sector Number last operated on by controller

T = Track Number programmed in Command Phase

Command Description (Continued)

Data Mark is read, the sector is read normally. If a Regular Data Mark is found and the SK bit is set, the sector is not read, bit 6 of ST2 (Control Mark) is set, and the next sector is searched for. If a Regular Data Mark is found and the SK bit is not set, the sector is read, bit 6 of ST2 (Control Mark) is set, and the read terminates with a normal termination.

WRITE DATA

The Write Data command is very similar to the Read Data command except that data is transferred from the μP to the disk rather than the other way around. If the controller detects the Write Protect signal, bit 1 of ST1 (Not Writable) is set and an abnormal termination is indicated.

WRITE DELETED DATA

This command is the same as the Write Data Command except a Deleted Data Mark is written at the beginning of the Data Field instead of the normal Data Mark.

READ A TRACK

This command is similar to the Read Data command except for the following. The controller starts at the index hole and reads the sectors in their physical order, not their logical order.

Even though the controller is reading sectors in their physical order, it will still perform a comparison of the header ID bytes with the Data programmed in the Command Phase. The exception to this is the sector number. Internally, this is initialized to a one, and then incremented for each successive sector read. Whether or not the programmed Address Field matches that read from the disk, the sectors are still read in their physical order. If a header ID comparison fails, bit 2 of ST1 (No Data) is set, but the operation will continue. If there is a CRC error in the Address Field or the Data Field, the read will also continue.

The command will terminate when it has read the number of sectors programmed in the EOT parameter.

READ ID

This command will cause the controller to read the first Address Field that it finds. The Result Phase will contain the header bytes that are read. There is no data transfer during the Execution Phase of this command. An interrupt will be generated when the Execution Phase is completed.

FORMAT A TRACK

This command will format one track on the disk. After the index hole is detected, data patterns are written on the disk including all gaps, address marks, Address Fields, and Data Fields. The exact details of the number of bytes for each field is controlled by the parameters given in the Format A Track command, and the IAF (Index Address Field) bit in the Mode command. The Data Field consists of the Fill Byte specified in the command, repeated to fill the entire sector.

To allow for flexible formatting, the μP must supply the four Address Field bytes (track, head, sector, number of bytes) for each sector formatted during the Execution Phase. In other words, as the controller formats each sector, it will request four bytes through either DMA requests or interrupts. This allows for non-sequential sector interleaving. Some typical values for the programmable GAP size are shown in Table X.

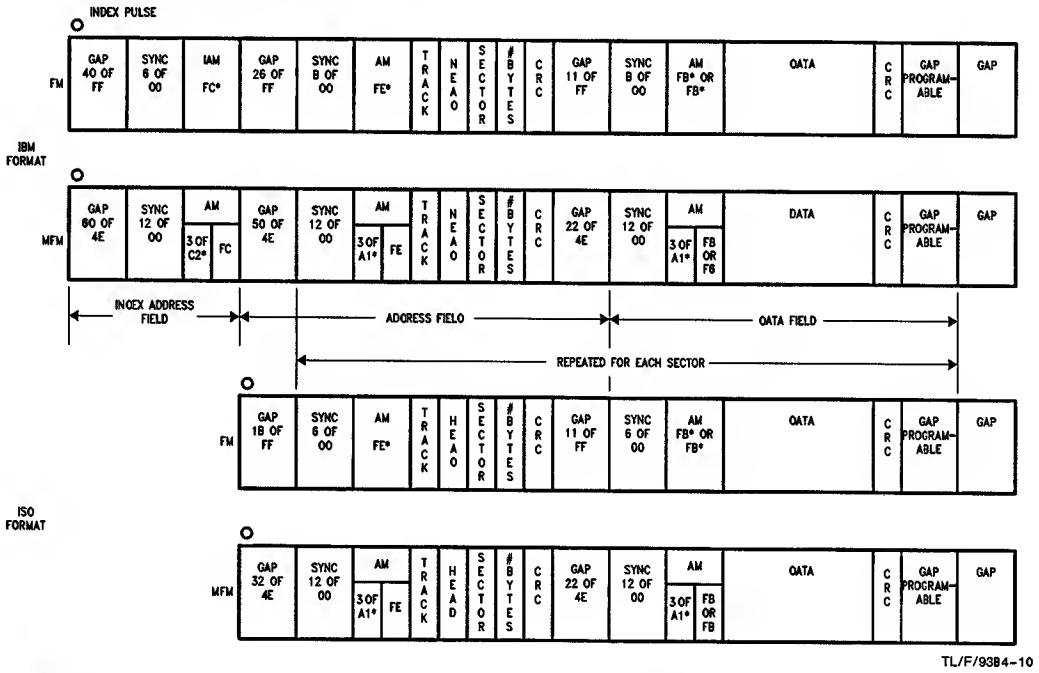
The Format Command terminates when the index hole is detected a second time, at which point an interrupt is generated. Only the first three status bytes in the Result Phase are significant.

TABLE X. Gap Length for Various Sector Sizes and Disk Types

Mode	Sector Size	Sector Code	EOT	Gap	Format Gap
8" Drives (360 RPM, 500 kb/s)					
FM	128	00	1A	07	1B
	256	01	0F	0E	2A
	512	02	08	1B	3A
	1024	03	04	47	8A
	2048	04	02	C8	FF
	4096	05	01	C8	FF
MFM	256	01	1A	0E	36
	512	02	0F	1B	54
	1024	03	08	35	74
	2048	04	04	99	FF
	4096	05	02	C8	FF
	8192	06	01	C8	FF
5.25" Drives (300 RPM, 250 kb/s)					
FM	128	00	12	07	09
	128	00	10	10	19
	256	01	08	18	30
	512	02	04	46	87
	1024	03	02	C8	FF
	2048	04	01	C8	FF
MFM	256	01	12	0A	0C
	256	01	10	20	32
	512	02	08	2A	50
	1024	03	04	80	F0
	2048	04	02	C8	FF
	4096	05	01	C8	FF
3.5" Drives (300 RPM, 250 kb/s)					
FM	128	00	0F	07	1B
	256	01	09	0E	2A
	512	02	05	1B	3A
MFM	256	01	0F	0E	36
	512	02	09	1B	54
	1024	03	05	35	74

Note: Format Gap is the gap length used only for the Format command.

Command Description (Continued)



TL/F/9384–10

Notes:

FE* = Data pattern of FE, Clock pattern of C7
FC* = Data pettern of FC, Clock pattern of D7
FB* = Date pettern of FB, Clock pattern of C7
F8* = Data pattern of F8, Clock pattern of C7
A1* = Date pattern of A1, Clock pattern of 0A
C2* = Data pattern of C2, Clock pettern of 14

All byte counts in decimal.
All byte values in hex.
CRC uses standard polynomial $x^{16} + x^{12} + x^5 + 1$.

FIGURE 6. IBM and ISO Formats Supported by the Format Command

SCAN COMMANDS

The Scan Commands allow data read from the disk to be compared against data sent from the μP. There are three Scan Commands to choose from:

Scan Equal	Disk Data = μP Data
Scan Less Than or Equal	Disk Data ≤ μP Data
Scan Greater Than or Equal	Disk Data ≥ μP Data

Each sector is interpreted with the most significant bytes first. If the Wildcard mode is enabled from the Mode command, an FF(hex) from either the disk or the μP is used as a don't care byte that will always match equal. After each sector is read, if the desired condition has not been met, the next sector is read. The next sector is defined as the current sector number plus the Sector Step Size specified. The Scan command will continue until the scan condition has been met, or the End of Track Sector Number has been reached, or if TC is asserted.

If the SK bit is set, sectors with deleted data marks are ignored. If all sectors read are skipped, the command will terminate with D3 of ST2 set (Scan Equal Hit). The result phase of the command is shown in Table XI.

TABLE XI. Scan Command Termination Values

Command	Status Register 2		Conditions
	D2	D3	
Scan Equal	0	1	Disk = μP
	1	0	Disk ≠ μP
Scan Low or Equal	0	1	Disk = μP
	0	0	Disk < μP
	1	0	Disk > μP
Scan High or Equal	0	1	Disk = μP
	0	0	Disk > μP
	1	0	Disk < μP

5

Command Description (Continued)

SEEK

There are two ways to move the disk drive head to the desired track number. Method One is to enable the Implied Seek Mode. This way each Individual Read or Write command will automatically move the head to the track specified in the command.

Method Two is using the Seek Command. During the Execution Phase of the Seek Command, the track number to seek to is compared with the present track number and a step pulse is produced to move the head one track closer to the desired track number. This is repeated at the rate specified by the Specify Command until the head reaches the correct track. At this point an Interrupt is generated and a Sense Interrupt Command is required to clear the interrupt.

During the Execution Phase of the Seek Command the only indication via software that a Seek Command is in progress is bits 0–3 (Drive Busy) of the Main Status Register. Bit 4 of the Main Register (Controller Busy) is not set. While the internal microengine is capable of multiple seeks on 2 or more drives at the same time since the drives are selected via the Drive Control Register in software, software should ensure that only one drive is seeking at one time. No other command except the Sense Interrupt Command should be issued while a Seek Command is in progress.

If the extended track range mode is enabled, a fourth byte should be written in the Command Phase to indicate the four most significant bits of the desired track number. Otherwise, only three bytes should be written.

RECALIBRATE

The Recalibrate Command is very similar to the Seek Command. It is used to step a drive head out to track zero. Step pulses will be produced until the track zero signal from the drive becomes true. If the track zero signal does not go true before 77 step pulses are issued, an error is generated. If the extended track range mode is enabled, an error is not generated until 3917 pulses are issued.

Recalibrations on more than one drive at a time should not be issued for the same reason as explained in the Seek Command. No other command except the Sense Interrupt Command should be issued while a Recalibrate Command is in progress.

SENSE INTERRUPT STATUS

An interrupt is generated by the controller when any of the following conditions occur:

1. Upon entering the Result Phase of:
   a. Read Data Command
   b. Read Deleted Data Command
   c. Write Data Command
   d. Write Deleted Data Command
   e. Read a Track Command
   f. Read ID Command
   g. Format Command
   h. Scan Commands
2. During data transfers in the Execution Phase while in the Non-DMA mode
3. Internal Ready signal changes state (only occurs immediately after a hardware or software reset).
4. Seek or Recalibrate Command termination

An interrupt generated for reasons 1 and 2 above occurs during normal command operations and are easily discernible by the μP. During an execution phase in Non-DMA Mode, bit 5 (Execution Mode) in the Main Status Register is set to 1. Upon entering Result Phase this bit is set to 0. Reasons 1 and 2 do not require the Sense Interrupt Status command. The Interrupt is cleared by reading or writing information to the data register.

Interrupts caused by reasons 3 and 4 are identified with the aid of the Sense Interrupt Status Command. This command resets the interrupt when the command byte is written. Use bits 5, 6 and 7 of ST0 to identify the cause of the Interrupt as shown in Table XII.

TABLE XII. Status Register 0 Termination Codes

Status Register 0			Cause
Interrupt Code		Seek End	
D7	D6	D5	
1	1	0	Internal Ready Went True
0	0	1	Normal Seek Termination
0	1	1	Abnormal Seek Termination

TABLE XIII. Step, Head Load and Unload Timer Definitions (500 kb/s MFM)

Timer	Mode 1		Mode 2		Unit
	Value	Range	Value	Range	
Step Rate	(16 − N)	1–16	(16 − N)	1–16	ms
Head Unload	N × 16	0–240	N × 512	0–7680	ms
Head Load	N × 2	0–254	N × 32	0–4064	ms

Issuing a Sense Interrupt Status Command without an interrupt pending is treated as an invalid command.

If the extended track range mode is enabled, a third byte should be read in the Result Phase which will indicate the four most significant bits of the Present Track Number. Otherwise, only two bytes should be read.

SPECIFY

The Specify Command sets the initial values for each of the three internal timers. The timer programming values are shown in Table XIII.

The Head Load and Head Unload timers are artifacts of the μPD765A. These timers determine the delay from loading the head until a read or write command is started, and unloading the head sometime after the command was completed. Since the DP8473's head load signal is now the software controlled Motor lines in the Drive Control Register, these timers only provide some delay from the initiation of a command until it is actually started. These times can be extended by setting the TMR bit in the Mode Command.

The Step Rate Time defines the time interval between adjacent step pulses during a Seek, Implied Seek, or Recalibrate Command.

The times stated in the table are affected by the Data Rate. The values in the table are for 500 kb/s MFM (250 kb/s FM) and 1 Mb/s MFM (500 kb/s FM). For a 300 kb/s MFM data rate (150 kb/s FM) these values should be multiplied by 1.6667, and for 250 kb/s MFM (125 kb/s FM) these values should be doubled.

The choice of DMA or Non-DMA operation is made by the NON-DMA bit. When this bit is 1 then Non-DMA mode is selected, and when this bit is 0, the DMA mode is selected.

This command does not generate an Interrupt.

Command Description (Continued)

LOW PWR (LOW PoWeR mode)

SENSE DRIVE STATUS

This two byte command obtains the status of a disk drive. Status Register 3 is returned in the result phase and contains the drive status. This command does not generate an interrupt.

MODE

This command is used to select the special features of the controller. The bits for the command phase bytes are shown in the command description table, and their function is described below. The defaults after a hardware or software reset are shown by the "bullets" to the left of each item.

- **TMR=0 (motor TiMeR):** Timers for motor on and motor off are defined for Mode 1. (See Specify Command)
  TMR=1: Timers for motor on and motor off are defined for Mode 2. (See Specify Command)

LW PR (LoW PoweR)

- 00 Completely disable the low power mode. (default)
  01 Go into low power mode 500 ms after the head unload timer times out.
  10 Go into low power mode now.
  11 Not Used.
- **IAF=0 (Index Address Format):** The controller will format tracks with the Index Address Field included. (IBM Format)
  IAF=1: The controller will format tracks without including the Index Address Mark Field. (ISO Format)
- **IPS=0 (ImPlied Seek):** The implied seek bit in the command is ignored.
  IPS=1: The implied seek bit in the command is enabled so that if the bit is set in the command, a Seek will be performed automatically.
- **ETR=0 (Extended Track Range):** Header format is the IBM System 34 (double density) or System 3740 (single density).
  ETR=1: Header format is the same as above but there are 12 bits of track number. The MSB's of the track number are in the upper four bits of the head number byte.
- **WLD=0 (scan WiLD card):** An FF(hex) from either the μP or the disk during a Scan Command is interpreted as a wildcard character that will always match true.
  WLD=1: The Scan commands do not recognize FF(hex) as a wildcard character.
  Head Settle: Time allowed for head to settle after an Implied Seek. Time = N × 4 ms, (0 ms–60 ms). (Based on 500 kb/s and 1 Mb/s MFM data rates. Double for 250 kb/s.)

PU (PUMP Pulse Output): When set enables a signal that indicates when the Data Separator's charge pump is making a phase correction. This is a series of pulses. This signal is output on the PUMP/PREN pin when this bit is set.

This is intended as a test mode to aid in evaluation of the Data Separator. (Default mode is off)

RG (Read Gate): Like the PUMP output, when this bit is set it enables a pin (the DSKCHG pin) to act as an external Read Gate signal for the Data Separator. This is intended as a test mode to aid in evaluation of the Data Separator. (Default mode is off)

SET TRACK

This command is used to inspect or change the value of the internal Present Track Register. This could be useful for recovery from disk mis-tracking errors, where the real current track could be read through the Read ID command and then the Set Track Command can set the internal present track register to the correct value.

The first byte of the command contains the command opcode and the R/W bit. If the R/W bit is low, a track register is to be read. In this case, the result phase contains the value in the internal register specified, and the third byte of the command is a dummy byte.

If the R/W bit is high, data is written to a track register. In this case the 3rd byte of the command phase is forced into the specified internal register, and the result phase contains the new byte value written.

The particular track register chosen to operate on is determined by the least significant 3 bits of the second byte of the command. The two LSB's select the drive (DR1, DR0), and the next bit (MSB) determines whether the least significant byte (MSB=0) or the most significant byte (MSB=1) of the track register is to be read/written. When not in the extended track range mode, only the LSB track register need be updated. In this instance, the MSB bit is set to 0.

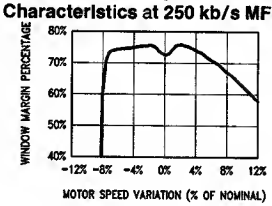
This command does not generate an interrupt.

INVALID COMMAND

If an invalid command (i.e., a command not defined) is received by the controller, the controller will respond with ST0 in the Result Phase. The Controller does not generate an interrupt during this condition. Bits 6 and 7 in the Main Status Register are both set to one's indicating to the processor that the Controller is in the Result Phase and the contents of ST0 must be read. When the system reads ST0 it will find an 80(hex) indicating an invalid command was received.

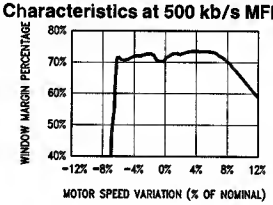
Typical Performance Characteristics

Typical Window Margin Performance Characteristics at 250 kb/s MFM



TL/F/9384–18

Typical Window Margin Performance Characteristics at 500 kb/s MFM



TL/F/9384–19

Absolute Maximum Ratings (Notes 1 and 2)

If Military/Aeroapace specified devicea are required, please contact the National Semiconductor Sales Office/Distributors for availability and apecifications.

Supply Voltage (V_{CC})	−0.5V to +7V
DC Input Voltage (V_{IN})	−0.5V to V_{CC} + 0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} + 0.5V
Storage Temperature Range (T_{STG})	−65°C to +165°C
Package Power Dissipation (P_D)	750 mW
Lead Temperature (T_L) (Soldering, 10 seconds)	260°C
$\lvert V_{CC}-V_{CCA}\rvert$	0.6V

Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
Operating Temperature (T_A)	0	+70	°C
ESD Tolerance: C_{ZAP} = 100 pF, R_{ZAP} = 1.5 kΩ (Note 5)	1500		V

DC Electrical Characteristics V_{CC} = 5V ±10% unless otherwise specified (Note 3)

Symbol	Parameter	Conditiona	Min	Max	Unita
V_{IH}	High Level Input Voltage	(except OSC2/CLK)	2.0		V
V_{IL}	Low Level Input Voltage	(except OSC2/CLK)		0.8	V
I_{IN}	Input Current (except OSC pins)	V_{IN} = V_{CC} or GND		±1.0	μA
I_{CCA}	Average V_{CCA} Supply Current	V_{IN} = 2.4V or 0.5V, I_O = 0 mA (Note 4)		20.0	mA
	Quiescent V_{CCA} Supply Current in Low Power Mode	V_{IN} = V_{CC} or GND, I_O = 0 mA (Note 4)		400	μA
I_{CC}	Average V_{CC} Supply Current	V_{IN} = 2.4V or 0.5V, I_O = 0 mA (Note 4)		20.0	mA
	Quiescent V_{CC} Supply Current in Low Power Mode	V_{IN} = V_{CC} or GND, I_O = 0 mA (Note 4)		2	mA
OSCILLATOR PINS (OSC2/CLK)					
I_{OSC}	OSC2 Input Current (OSC1 = GND)	V_{IN} = V_{CC} or GND	±1.6		mA
V_{IH}	OSC2 High Level Input Voltage	OSC1 = GND	2.4		V
V_{IL}	OSC2 Low Level Input Voltage	OSC1 = GND		0.4	V
MICROPROCESSOR INTERFACE PINS (D0–D7, INT, DAK, TC, DRQ, RD, WR, CS, A0–A3)					
V_{OH}	High Level Output Voltage	I_{OUT} = −20 μA I_{OUT} = −4.0 mA	V_{CC} − 0.1 3.5		V V
V_{OL}	Low Level Output Voltage	I_{OUT} = 20 μA I_{OUT} = 12 mA		0.1 0.4	V V
I_{OZ}	Output TRI-STATE® Leakage Current	V_{OUT} = V_{CC} or GND		±10.0	μA
DISK DRIVE INTERFACE PINS (MTR0–3, DR0–3, WDATA, WGATE, RDATA, DIR, HDSEL, TRK0, WRTPRT, RPM, STEP, DSKCHG, INDEX)					
V_H	Input Hysteresis		250 Typical		mV
V_{OL}	Low Level Output Voltage	I_{OUT} = 48 mA		0.4	V
I_{LKG}	Output High Leakage Current	V_{OUT} = V_{CC} or GND		±100	μA
V_{IH}	High Level Input Voltage		2.2		V
V_{IL}	Low Level Input Voltage			0.8	V

Note 1: Absolute Maximum Retings ere those values beyond which demege to the device may occur.

Note 2: Unless otherwise apecified ell voltagea are referenced to ground.

Note 3: These DC Elactrical Charecteristics ara measured ststicly, and not undar dynamic conditiona.

Note 4: I_{CC} is measured with a 0.1 μF supply dacoupling capacitor to ground.

Note 5: Value basad on test complying with NSC SOP5-028 human body model ESD tasting using the ETS-910 tester.

Phase Locked Loop Characteristics V_{CC} = 5V ±10%, F_{XTAL} = 24 MHz unless otherwise specified

Symbol	Parameter	Conditions	Typ		Units
V_{REF}	SETCUR Pin Reference Voltage	R_1 = 5.6 kΩ, V_{CC} = 5V	1.1		V
K_{VCO}	VCO Gain (Note 5)	t_{DATA} = 1 μs ±10%	25		Mrad/s/V
R_1	Recommended Pump Resistor Range		3–12		kΩ
$K_{P(UP)}$	Charge Pump Up Current Gain ($I_{REF}/I_{P(UP)}$) (Note 6)	R_1 = 5.6 kΩ	2.50		(none)
$K_{P(DWN)}$	Charge Pump Down Current Gain ($I_{REF}/I_{P(DWN)}$) (Note 6)	R_1 = 5.6 kΩ	2.25		(none)
K_{PLL}	Internal Phase Locked Loop Gain (Note 7)	(R_1 = 5.6 kΩ) Pump Up Pump Down	 75 70		 Mrad Mrad
T_{SW}	Static Window (Note 8)		**Early**	**Late**	
		(R_1 = 5.6 kΩ) 250 kb/s 500 kb/s 1.0 Mb/s	 1075 530 259	 872 440 234	 ns ns ns
T_{DW}	Dynamic Window Margin	(Note)	70		%

Note: Measuremente made with e repeetlng 'DB6" date pettern with reverse writa precompensetion, using recommended filter values for the configuration shown in Figure 4c. 25°C, 5.0V, 0% MSV.

Note 5: The VCO gein is measured at the 1.0 Mb/s data rata by forcing the data period over e renge from 900 ns to 1100 ns, and measuring the rasulting voitage on the filter pin. The best straight line gein is fit to tha measured points.

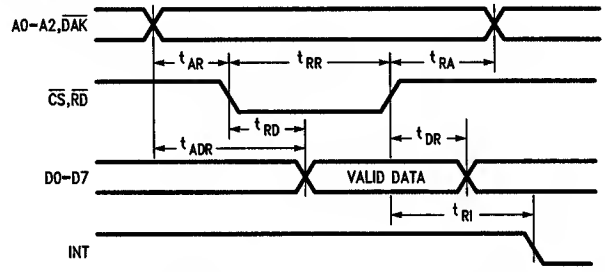
Note 6: Thia is tha current gain of the charge pump, which is defined as the output current divided by tha current through R_1.

Note 7: This is the product of: $V_{REF} \times K_P \times K_{VCO}$. The total variation in this specification indicates the total loop gein variation contributed by the internal circuitry. The K_{VCO} portion of this epecification is measured at the 1.0 Mb/s data rata by forcing the dete period over e range of 900 ns to 1100 ns, and measuring the reeultant K_{VCO}. K_P is measured by forcing the Filter pin to 2.1V and measuring the ratio of the charge pump current over the input current.

Note 8: The DP6473 is guaranteed to correctly decoda a single ehifted ciock puise at the end of e long seriea of non-shifted preamble bits es long es the singie shifted pulse la shifted lese than the amount specified in T_{SW}. The length of the preambie is long enough for tha PLL to lock. The filter components used are those in Table II.

AC Electrical Characteristics

MICROPROCESSOR READ TIMING

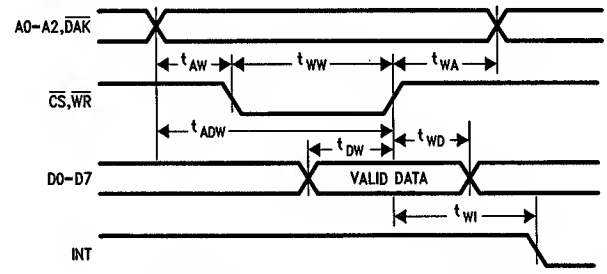


TL/F/9384–11

Symbol	Parameter	Min	Max	Units
t_{AR}	Address Valid prior to Read Strobe	10		ns
t_{RA}	Address Hold from Read Strobe	0		ns
t_{RR}	Read Strobe Width	75		ns
t_{RD}	Read Strobe and Chip Select to Data Valid		75	ns
t_{ADR}	Address Valid to Read Data		85	ns
t_{DR}	Data Hold from Read Strobe to High Impedance (TRI-STATE Note)	5	60	ns
t_{RI}	Clear INT from Read Strobe		65	ns

TRI-STATE Note: This limit includes the RC delay inherent in our test method. This signal will typically turn off within 15 ns, enabling other devices to drive this signal with no contention.

MICROPROCESSOR WRITE TIMING

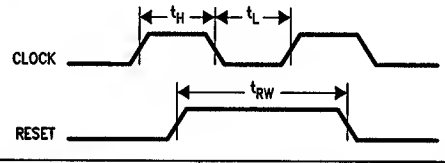


TL/F/9384–12

Symbol	Parameter	Min	Max	Units
t_{AW}	Address Valid to Leading Edge of Write Strobe	10		ns
t_{WA}	Address Hold from Write Strobe	0		ns
t_{WW}	Write Strobe Width	25		ns
t_{ADW}	Address Valid to Trailing Edge of Write Strobe	35		ns
t_{DW}	Data Setup to End of Write Strobe or Chip Select	20		ns
t_{WD}	Data Hold from Write Strobe	12		ns
t_{WI}	Clear INT from Write Strobe		65	ns

AC Electrical Characteristics (Continued)

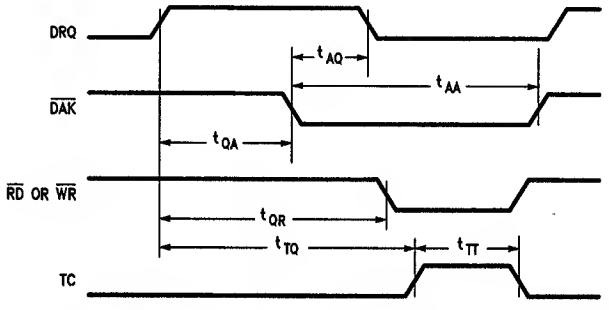
OSC2/CLOCK AND RESET TIMING



TL/F/9384–13

Symbol	Parameter	Min	Max	Units
t_H	Clock High Time	16		ns
t_L	Clock Low Time	16		ns
t_{RW}	Reset Pulse Width	100		ns

DMA TIMING (Note 9)



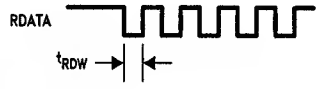
TL/F/9384–14

Symbol	Parameter	Min	Max	Units
t_{AQ}	End of DRQ from DAK		115	ns
t_{QA}	DAK Assertion from DRQ	10		ns
t_{AA}	DAK Pulse Width	75		ns
t_{QR}	DRQ to Read or Write Strobe	10		ns
t_{TT}	TC Strobe Width	50		ns
t_{TQ}	Time after Last DRQ That TC Must Be Asserted By		(Note 10)	ns

Note 9: DMA Acknowledge is sufficient to acknowledge a dets transfer. Rsad or Write Strobes srs neccessary only if dsts is to be presented to ths data bus. If Resd/Write Strobes sre spplied, then they snd the Acnowledge must be removed within 1 μs of each other.

Note 10: TC is is the terminsl count pin which terminstes the data transfer operation. There are sevsral constraints pleced on the timing of TC. 1) TC is snabled by $\overline{DAK}$, so TC must be pulsed whils $\overline{DAK}$ is low. 2) TC must occur befors ((1/data rate x 8) − 1 μs). Data rste is the exsct dsta transfer rate being used.

DRIVE READ TIMING

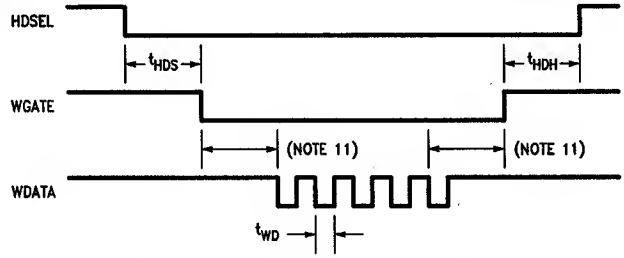


TL/F/9384–15

Symbol	Parameter	Min	Max	Units
t_{RDW}	Read Data Pulse Width	25		ns

AC Electrical Characteristics (Continued)

DRIVE WRITE TIMING

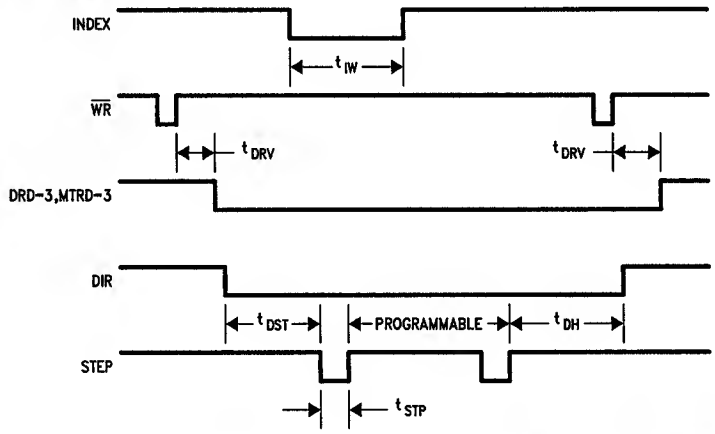


TL/F/9384–16

Symbol	Parameter	Conditions	Min	Max	Units
t_{WD}	Write Data Pulse Width	250 kb/s (MFM)	500		ns
		300 kb/s (MFM)	416		ns
		500 kb/s (MFM)	250		ns
		1000 kb/s(MFM)	225		ns
t_{HDS}	Head Select Setup to Write Gate Assertion		50		μs
t_{HDH}	Head Select Hold from Write Gate		15		μs

Note 11: Whenever WGATE is esserted the WDATA line is ective. At the end of each write one dummy byte is written before WGATE is deesserted.

DRIVE TRACK ACCESS TIMING



TL/F/9384–17

Symbol	Parameter	Min	Max	Units
t_{DST}	Direction Setup prior to Step	6		μs
t_{DH}	Direction Hold from End of Step	1 step time		
t_{STP}	Step Pulse Width	8		μs
t_{IW}	Index Pulse Width	100		ns
t_{DRV}	Drive Select or Motor Time from Write Strobe		100	ns

AC Test Conditions (Notes 11, 12, 13)

Input Pulse Levels	GND to 3V
Input Rise and Fall Times	6 ns
Input and Output Reference Levels	1.3V
TRI-STATE Reference Levels	Active High − 0.5V Active Low + 0.5V

Note 11: C_L = 100 pF, includes jig and scopa capacitsnce.

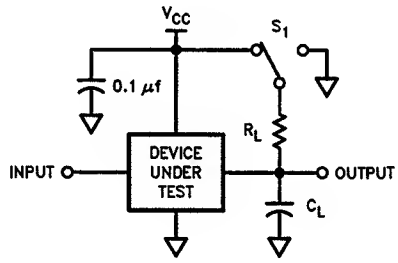
Note 12: S1 = open for push-pull outputs. S1 = V_{CC} for high impedanca to active low and activa low to high impedanca measuremants. S1 = GND for high impedanca to active high and active high to high imepedance measuramants. R_L = 1.0 kΩ for μP interfaca pins.

Note 13: For tha Open Drain Driva Interface Pins S1 = V_{CC} and R_L = 150Ω.

Capacitance T_A = 25°C, f = 1 MHz (Note 14)

Symbol	Parameter	Typ	Units
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	8	pF

Note 14: This parametar is not 100% testad.



TL/F/9384–20

Floppy Disk Data Separator Design Guide for the DP8473

National Semiconductor
Application Note 505
Bob Lutz, Paolo Melloni
and Larry Wakeman

Table of Contents

1.0 INTRODUCTION

Due to the increase in CMOS processing capabilities it is now possible to integrate both the analog and digital circuitry to achieve a high performance monolithic data separator. The choice of CMOS technology also enables the integration of an analog data separator function with good performance onto a floppy disk controller, resulting in National's DP8473 integrated floppy data separator/controller.

This paper discusses the functionality of the DP8473 data separator blocks, after a brief introduction to floppy disk data separator theory. It then delves into the detail of PLL design theory, providing design equations and considerations that enables the user to optimize the performance of the PLL for various applications.

2.0 THE FUNCTION OF A DATA SEPARATOR

2.1 Encoding Techniques

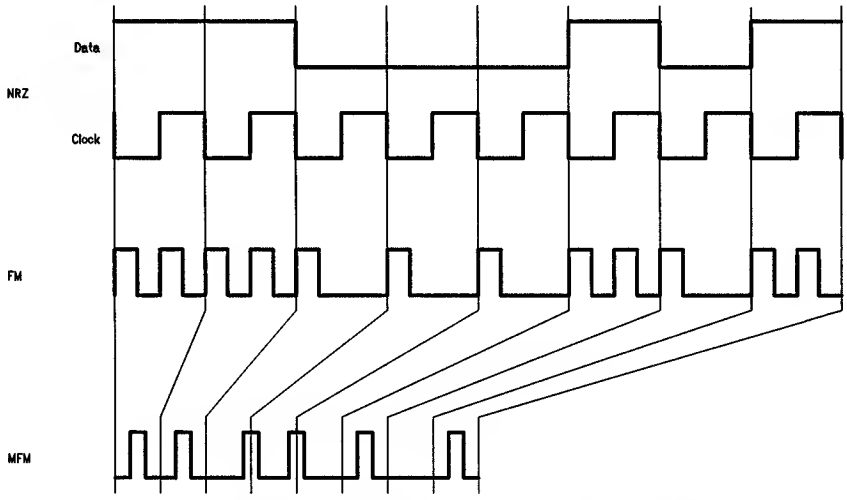
The floppy disk controller writes data to the floppy disk drive in a bit serial fashion as a series of encoded pulses. These pulses are then converted by the drive into magnetic flux reversals on the floppy disk media. The pulses can be later read by the drive and converted back to encoded pulses which can be decoded by the controller into the original data.

Since data is one serial set of bits, and because "real world" imperfections in the writing/reading process can cause the serial information to vary and jitter, the clocking information is embedded into the data stream, which enables synchronization to the data by the circuitry in charge of reading the data.

It is the purpose of the Data Separator circuit to take the encoded data from the disk, and to recover and separate out the clock signal. The separated clock and data signals are then sent to the controller's deserializer which converts the data to bytes of data suitable for microprocessor manipulation.

The two most popular encoding schemes used on floppy disks are: FM (Frequency Modulation), and MFM (Modified Frequency Modulation). FM defines a bit cell for each bit of data. Each cell contains a position for a clock pulse and a position for a data pulse. Each of these positions are referred to as windows. The clock pulse is present in every cell and a data pulse is present only if the data bit for that cell is a one. When this data is read back from a disk, a read clock can be generated from the clock pulses of the signal. An example of FM encoded data is shown in *Figure 1.*

FM encoding was the first method used for recording data on a floppy disk. It is still used in some low cost systems where storage capacity is not a critical issue. This method works very well and requires relatively simple circuitry to separate the clock pulses from the data pulses when the data is read back. However, only 50% of the useful disk space is used for recording data. The other 50% is used to record clock pulses.



TL/F/9419–1

FIGURE 1. Examples of how clock and data information is encoded into FM and MFM formats. Notice the increased density of MFM over FM.

MFM encoding allows 100% of the useful disk space for storing data, and is currently the most widely used recording format used for floppy disks. MFM defines a bit cell for each bit of data, similar to FM. Again, each cell contains a position for a clock pulse (clock window) and a position for a data pulse (data window). A data pulse is present if the data bit is a one. A clock pulse is present only if the data bit is a zero and the data bit in the previous bit window was a zero.

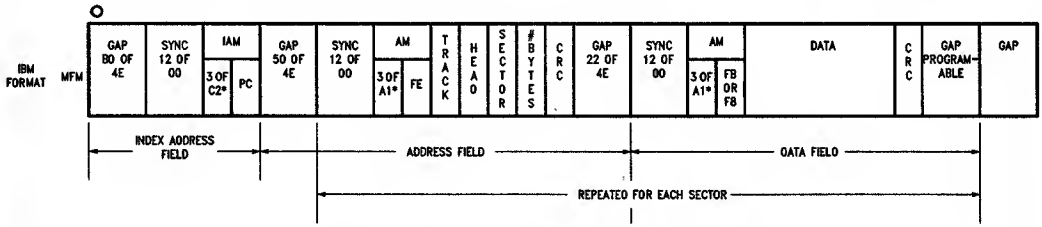
A comparison of FM and MFM can be seen in *Figure 1*. Because MFM requires fewer pulses to encode the same amount of data, the information can be stored in half the area required for FM encoded data. The only drawback of MFM is that it requires better read/write head and accompanying electronics. It also requires a higher precision data separator than FM requires. This is to resolve the location of each pulse more precisely than with FM.

2.2 Typical Floppy Format

A disk consists of many separate tracks. These tracks are configured as a set of concentric circles. Each track contains a set of many sectors. Each sector contains one Address Field and one Data Field. *Figure 2* shows the most common track format used on floppy disks today. It is the IBM double density standard.

The Address Field within a sector is used to identify what sector the following data field belongs to. It begins with a synchronization field or preamble to allow the data separator (which will be described soon) to synchronize to the speed at which the data is being read from the disk.

Next an address mark uniquely identifies this field as an Address Field. The address marks are encoded with a unique illegal pattern that is an MFM encoding rule violation. The violation is a missing clock pulse from a particular location within the byte. This illegal pattern guarantees that this is an address mark field and not a data pattern from some other area of the disk. The following four bytes identify the sector being read. This is followed by a CRC (Cyclic Redundancy Check). The CRC allows the controller to verify that the information read is free of errors. This is followed by a gap which is simply a series of bytes that physically separates the Address Field from the Data Field.



TL/F/9419–2

FIGURE 2. Typical format of a floppy disk. This is the IBM MFM standard.

Notes:

C2* = Data Pattern of C2, Clock Pattern of 14
A1* = Data Pattern of A1, Clock Pattern of 0A

The Data Field contains the data that the sector represents. It begins with a preamble and data field address mark, similar to the beginning of the Address Field. The actual sector data follows this. The data is followed by a CRC and then a gap, that separates this sector from the next.

2.3 Obstacles in Reading Data

Since MFM is the most popular floppy disk data encoding method the following discussion will refer specifically to MFM.

The floppy controller must be able to decode the data read from a disk drive. Theoretically, this could be a fairly easy process. The controller must first synchronize to the clock pulses in the preamble field of a sector. After that, it is just a matter of checking when the next encoded data pulse arrives. The pulse can arrive, 1, 1.5 or 2 bit periods later. Using this information the controller can decode this and all subsequent bits, reconstructing the original data from this information.

Unfortunately, this simple method is not so simple. The data pulses read back from a disk drive will generally be somewhat different from the data originally written.

There are three major sources of data degradation.

1. **Bit Shift**—As data is written, magnetic interaction of adjacent bits cause the data to be shifted in time from its nominal position. When these flux transitions are recorded close to each other, the superposition of their magnetic fields tends to move their apparent position. Thus when they are read, the floppy drive's peak detector moves the peak of these flux transitions apart from each other. This is the major cause of instantaneous bit shift.

   (This type of data degradation is mostly predictable and can be partially compensated for by shifting the data as it is written to the disk in the opposite direction that the bit is predicted to shift. This is called Write Precompensation. The DP8473 contains circuitry required to perform this function. The write precompensation circuit intercepts the serial data being written to the disk and shifts the data early, late, or none, based on the data pattern.)

   Several other factors can contribute to jitter. The drive's peak detector may be unbalanced, resulting in a bit shift similar to that described above. This could cause positive going peaks to appear earlier than negative going peaks or vice-versa. Also, a long cable between the disk drive and the disk controller may contribute to bit shift.

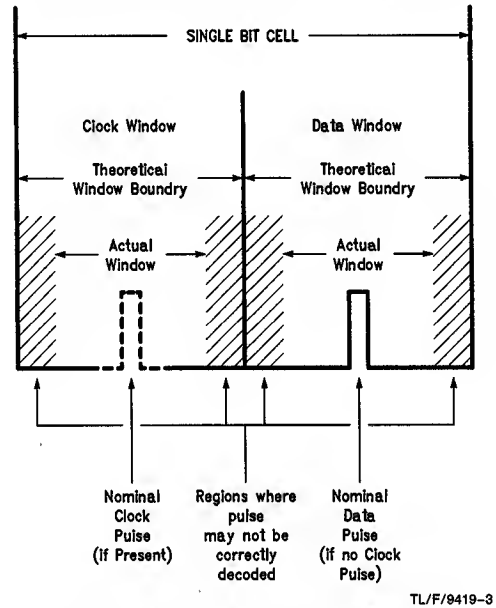
2. **Motor Speed Variation (MSV)**—This is an error in the spindle motor speed from the nominal, and causes the data rate to vary typically 1–2% for each drive. For design purposes this value is doubled since drive media is interchangeable. Thus a slow drive can record data that is read on a fast drive.

3. **Instantaneous Speed Variation (ISV)**—This is an additional speed error that is a constantly changing affect due to disk-jacket friction, and mechanical resonances. Usually this variation has a frequency component less than 1 kHz and causes the data rate to vary an additional 1–2% (again doubled).

While bit shift is the primary cause of decoding problems, the speed variations create difficulty in locking to the frequency of the data stream, and degrade jitter tolerance. The data separator must be able to synthesize the average frequency of the data coming in, and if the disk data rate differs from the nominal value then synchronization is more difficult.

2.4 Performance Measures of a Data Separator

There are several measures of data separator performance. The most universal one is window margin. Window margin measurements themselves can be subdivided into two categories, Dynamic, and Static (described shortly). Window margin is defined as the amount of bit shift that can be tolerated by a data separator without mis-decoding the data.



TL/F/9419–3

FIGURE 3. Window Margin Timing Definition

Figure 3 shows a timing diagram of a typical bit cell, and its composite data and clock windows. Theoretically a pulse (either clock or data) could be shifted in time either early or late relative to its nominal position by up to ¼ of a bit period and still be decoded correctly. This is the theoretical window boundary region in *Figure 3*. If the pulse is shifted more than this, then it would fall into another pulse's window. In reality, due to the limitations of practical data separator implementations, the actual window boundary in which data will be directly decoded is less than the full ¼ period. This is shown in *Figure 3* as the actual window region. Typically this actual window size is measured as either a percentage of the theoretical maximum or in terms of nanoseconds. The former is the more popular method and will be used here as well. In equation form:

$$WM\% = \frac{\text{(Actual Window Size)}}{\text{(Nominal Theoretical Window Size)}} \times 100\%$$

Window margin should be measured with a specified amount of ISV and MSV, at a specific data rate, with a specified data field and format pattern, and a known bit shift algorithm. If any of these are unspecified, then a true comparison of data separator performance is more difficult. Window margin is usually specified as a percentage of the nominal frequency window (as opposed to the nominal frequency plus the MSV).

There are two basic types of window margin tests. One test is where the data separator and controller must read a sector of data, in which all the bits are shifted until the data separator cannot read the sector correctly. This is called dynamic window margin. A second test is to present a long sequence of perfectly centered MFM clock pulses except for one bit. This bit is shifted until an error occurs. This second test is called static window margin.

Do Not Confuse the Two Measurements. The first one more correctly reflects the "Real World". The second one is an indicator of the accuracy of the circuits that compose the PLL, but does not include most of the errors due to the response of the PLL.

As an example of a dynamic window margin test: A data separator has a 70% window margin at 500 Kb/s, with a total ±1.5% MSV, and ±1% ISV. The encoding is MFM, and the data pattern is a repeating DB6DB6 . . . (HEX) pattern. A reverse write precompensation algorithm is used for pattern dependent bit jitter (all bits are jittered). These conditions are one of the worst case conditions for analog data separators. This means that the data separator will correctly decode a pulse so long as it is shifted no more than ±350 ns from its nominal position (the theoretical window at 500 Kb/s is ±500 ns) over the full MSV and ISV range.

Another data separator performance measurement is Bit Error Rate (BER). This is a measure that is defined as ratio of the number of bit errors during long term reading divided by the total number of bits read. A small bit error rate is desirable. BER is better for evaluation of total system performance, since the performance of the whole system effects on this figure. Thus as a final system checkout the manufacturer can specify the media, drives, data separator, and determine the error rate of this system. It is relatively difficult to isolate the bit errors due solely to the data separator. This specification is a less exact performance measurement than window margin for a data separator.

Why is Window Margin Important?

The greater the window margin the lower the error rate. For example if a bit is read with too much bit jitter for the data separator, then that data cannot be read and the whole sector (or file) is lost. This is especially fatal since floppies do not have error correction capability.

Another area where window margin is important, is manufacturing yields. A larger window margin ensures that when intermixing best/worst case drives and controllers there is minimum fallout. Thus larger volume vendors tend to try to optimize window margin to improve yields as well as data integrity.

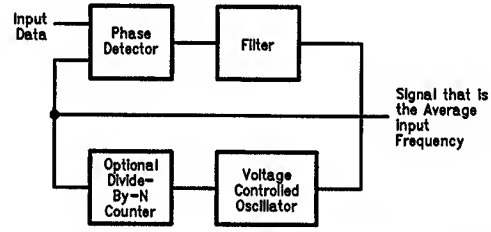
Each designer needs to decide for himself what margin requirements are necessary. In general, many high quality analog designs typically achieve 60–65% window margin under worst case conditions, with the best designs approaching 70%.

Later in sections 4.2 and 5.1 theoretical calculation and practical measurements of window margin are described.

2.5 Analog Data Separator Basics

The job of a data separator is to produce a read clock that follows the slow data rate change caused by the drive motor variation (MSV and ISV), but not track the instantaneous bit jitter. This read clock then is used to clock in the serial data into some type of deserializer. Generating a read clock for MFM encoded data is potentially difficult. Because of the MFM encoding rules, many clock pulses are missing from clock or data windows. As a matter of fact, in a long string of one's, there are no clock pulses at all. A data separator must use both clock pulses and data pulses to synchronize to the encoded signal. The most popular method to do this is with a Phase Locked Loop (PLL).

A PLL consists of three main components, a phase detector, a filter, and a voltage controlled oscillator (VCO), as shown in *Figure 4.* Also in most cases a divider is used to divide the VCO frequency as needed. The basic operation of a PLL is fairly straight-forward. The phase detector detects the difference between the phase of the VCO (or divider) output and the phase of a periodic input signal. This phase difference is converted to a current which either charges or discharges a filter. The resulting filter voltage changes the frequency of the VCO (and also the divider) output in an attempt to reduce the phase difference between the two phase detector input signals. A PLL is "locked" when the frequency of the two phase detector input signals is the same.



TL/F/9419–4

FIGURE 4. Simplified Block Diagram of Phase Locked Loop

With a slight modified version of the basic PLL, an MFM data separator can be made. A modification is required because MFM encoded data is not a periodic signal. A phase comparison can only be made when a pulse arrives from the disk. When there is no clock or data pulse, the PLL should continue generating the frequency it was generating before the missing pulse. This is called a phase only comparison, and it is the usual method of tracking an MFM signal.

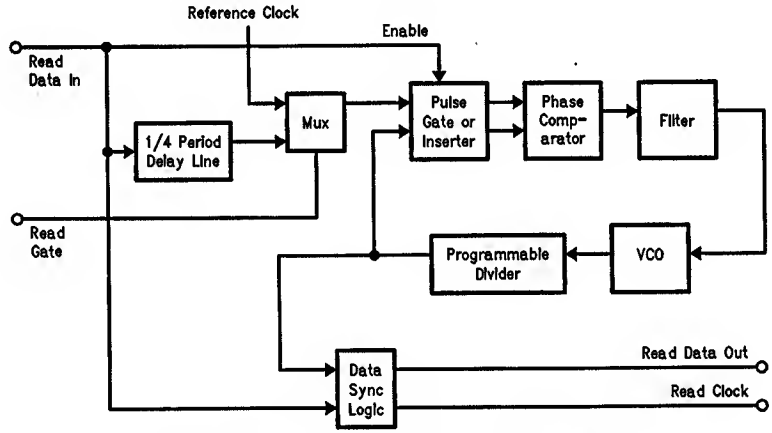


FIGURE 5. Simplified Block Diagram of Typical Data Separator

A typical data separator is shown in *Figure 5*. In addition to the components of a typical PLL, it includes a quarter period delay line and either a pulse gate or pulse inserter (not both). Both of these blocks enable the pulse gate or pulse inserter to decide when the phase comparison should be made. The quarter period delay line delays the incoming data pulses a quarter of a bit cell, and this feeds the pulse gate or pulse inserter. The pulse gate will disable phase comparisons when a VCO pulse occurs but read data pulses are missing. The pulse inserter will insert fake read data pulses into the phase detector when there is a VCO pulse but no read data pulse. These components are required to determine the proper timing of the phase comparisons for MFM encoded data. The need for these blocks can be demonstrated by referring to *Figure 6*. Only the use of the pulse gate is described since this is what is implemented in the DP8473 data separator.

Figure 6a shows two MFM bit cells, each with a clock pulse. The VCO output provides two clocks per cell since an MFM pulse can appear in either of the two windows that compose the bit cell. (Note for simplicity the divider block is ignored.) To achieve lock, the data separator tries to line up the rising edge of the input pulses with the rising edges of the VCO output cycles. MFM encoded data is not periodic, that is some of the cells are missing pulses. The data separator must decide when to make a valid phase comparison. This can be seen from *Figure 6a* where the phase detector first makes a comparison to an early pulse, which is correct, but then on the next VCO cycle the phase detector now compares this VCO edge even though no input pulses are present. Hence, there must be a mechanism for fooling the phase detector into not making a comparison. The method chosen in the DP8473 is to use a pulse gate to eliminate the unwanted VCO edge.

However, disabling the phase detector's input does not completely solve the problem, as shown in *Figure 6b*. Here the first early pulse is compared correctly. At the beginning the next VCO cycle the data separator does not know whether to do a phase comparison, since it does not know whether the pulse is missing or just late. Thus by the time the next pulse does arrive the PLL is lost.

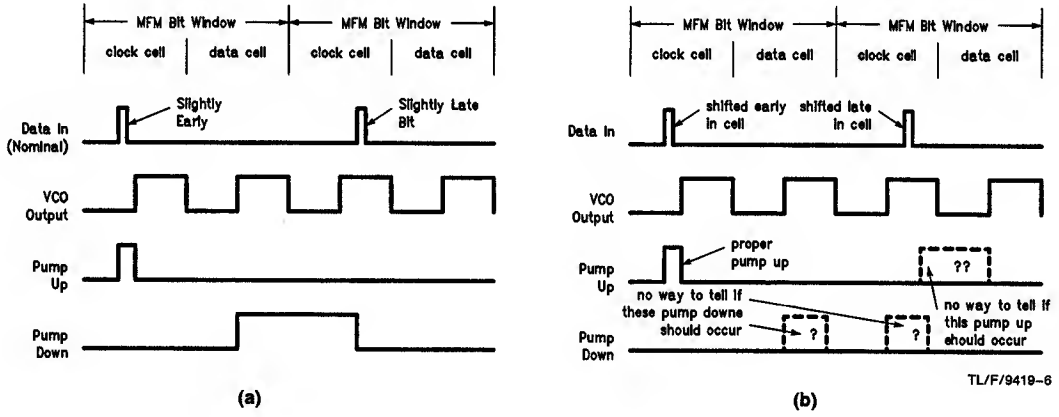
Therefore, the DP8473 data separator uses a ¼ period (½ bit window) long delay line with the pulse gate. Now with this delay line, all phase comparisons are made to the delayed data. Thus the PLL is operating ¼ of a period behind the data coming from the disk, but this allows the phase comparison enable logic to determine whether a pulse will occur in a bit cell or not, and make the proper comparison.

Figure 6c shows how this works. For an early bit, the data input enables a phase comparison, and the phase detector compares the delayed data bit to the VCO edge. In the case of this early bit, the proper pump up is generated. On the next VCO cycle, the quarter period delay has detected no pulse, and so no comparison is made. For the late bit, the comparison is enabled prior to the VCO clock, so a pump down is generated until the delayed data bit is seen by the phase detector.

At nominal frequency, a delay of ¼ of a bit ensures that the phase detector will be properly enabled even if the data bit is late all the way to the edge of its clock or data window. (Remember one bit cell contains a clock and a data window. A data pulse will appear within either (but not both) of these windows. Therefore the theoretical maximum amount of bit shift is ¼ of a bit cell.)

The quarter period delay line solves this problem of forecasting the future. It causes the MFM encoded data pulses to be delayed by a quarter of a bit period. This allows the pulse gate to determine when data pulses exist ahead of time and thus enable the phase detector only at the appropriate times.

It is important that the quarter period delay line be accurate. If the quarter period delay line is not accurate (ie. it's too long or too short), then the window margin performance of the data separator will be reduced. This performance reduction is due to the PLL's inability to correctly resolve bit shift near the edge of a bit window. For example, if at 500 Kb/s the delay line were shorter than it should be, say 400 ns long instead of 500 ns, then a bit shifted 450 ns from its nominal position is incorrectly decoded. The window margin in this case is immediately reduced 20% from it's ideal. The same degradation occurs when the delay line is too long.



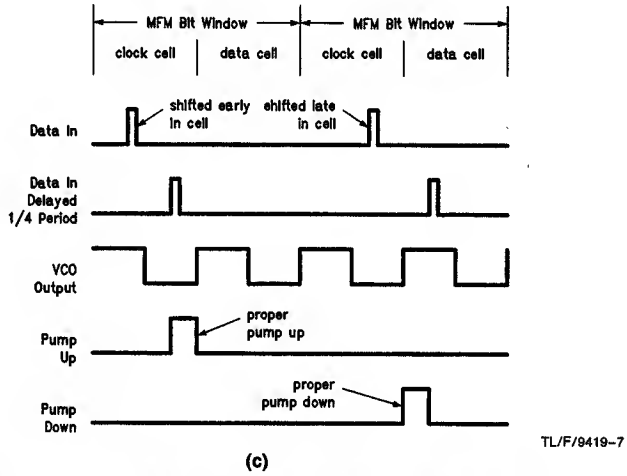


FIGURE 6. This ahowa why a ¼ period delay line is needed. a) Shows phaae comparisona that occur if only a phase detector is used; b) Shows the data separator's need to predict the arrival of a pulae; and c) Shows how the ¼ period delay fixea this.

2.6 Operation of an Analog Data Separator

A data separator can be described as operating in one of three phases during each read cycle: Idle Phase, Initial Locking Phase, and the Tracking Phase.

Initially, when the data separator is not being used to read data from the disk, it is in the Idle Phase. While in the Idle Phase, the PLL is both phase and frequency locked to a reference frequency. (Frequency comparison is implemented by forcing a phase comparison every VCO clock.) The PLL must eventually lock to both clock and data pulses of the encoded data when it is read from the disk, so the reference frequency is generally two times the data rate frequency.

When data is to be read from the disk, the PLL switches from the reference frequency to the incoming data stream. Because the encoded data read from the disk is not a periodic signal, only phase comparisons are made. Since the PLL was initially locked to a frequency very close to twice the actual data rate, the time required for the PLL to lock onto the data read from the disk is minimized.

To further minimize this locking time, the beginning of each Address Field and Data Field starts with a preamble (or synchronization field). The preamble is a series of bytes with a zero data pattern (all clock pulses and no data pulses). When read, the preamble will produce a periodic signal with little bit jitter. The data separator can lock to this signal with the least chance of an error. It would be ideal for the floppy controller to switch the data separator from the Idle Phase to the Initial Locking Phase at the beginning of a preamble to enable the maximum amount of lock time.

Once the PLL is locked to the average frequency of the data being read from the disk, it should simply track the data frequency. This means tracking the slow data rate speed variations caused by the drive motor, yet ignoring instantaneous bit jitter. This is the Tracking Phase. The data separator then allows the controller's deserializer to start decoding the incoming data.

2.7 Digital Data Separators

A second method of separating clock and data information is to use a digital data separator. While the circuits for the analog solution has evolved significantly, digital data separators have also improved somewhat, in a (less than successful) attempt to match the performance of the analog approach. These circuits are described below.

First Generation Digital Data Separator (DDS)—This circuit is a very convenient all digital data separator. Its primary advantages are simplicity, and low external parts count. This circuit usually consists of a set of counter timing circuits and some control logic to count times between individual pulses, and thus determine whether a pulse is clock or data. The SMC9216 is representative of this technology. Its major disadvantage is performance, and the inability to optimize the window margin for various lock ranges. The dynamic window margin for these types of circuits is usually around 55% with no MSV and as low as 30% with a ±3% total MSV.

Second Generation DDS—A sophisticated digital data separator can be compared functionally to an analog data separator. The ideal digital separator consists of a sampler (phase detector), a ROM look up table, with memory (filter), and a programmable counter (VCO). The pulse gate can be implemented as an extension of the ROM look up table. These circuits are typically much better than 1st generation circuits, but still far short of the analog approach. These circuits have dynamic window margins of 50–55% over a ±6% lock range (total MSV variation).

3.0 DP8473 DATA SEPARATOR FUNCTIONAL DESCRIPTION

The integrated floppy disk data separator from National Semiconductor combine the performance of an analog PLL and the ease of use of a digital data separator. It does not require any external trimmed components, and it has a data rate range from 125 Kbits/sec up through 1.0 Mbits/sec. It is built using CMOS technology to achieve good linear performance as well as low power operation. A block diagram for the data separator is shown in *Figure 7*.

3.1 Block Diagram Description

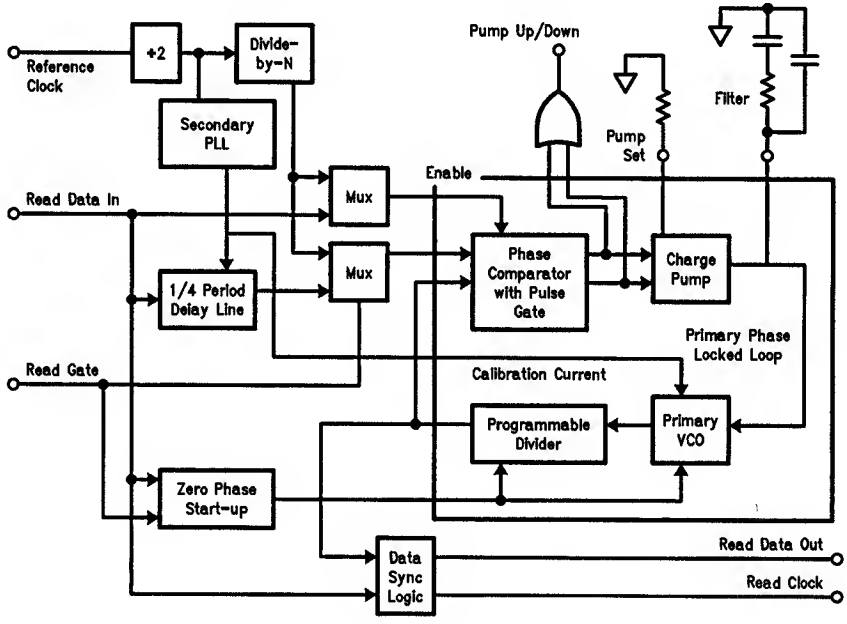
The heart of the DP8473 data separator is the main PLL. The main PLL consists of the VCO, programmable divider, phase detector, and the charge pump. The entire operation of the PLL and data separator logic is based on the Reference Clock which should be an accurate reference frequency. The Reference Clock is divided by two, then feeds the Secondary PLL, and the divide-by-N counter. As discussed later the Secondary PLL is used to calibrate the operation of the quarter period delay and Primary VCO. The Reference Clock's Divide-By-N counter and the Programmable Divider are both programmable counters whose divide by factor is determined by the data rate selected. The output of the divide-by-N and the Programmable divider is always twice the data rate. The output of the divide-by-N is used as a reference frequency for the PLL to lock to when the PLL is idle. The output of the Programmable Divider is the separated clock that is used to strobe the incoming pulses into the controller's deserializer.

Note: Throughout this discussion, the Reference Clock es shown in *Figure 7* is the master clock for the deta separator block. This Reference Clock elso generates several other clock frequencies that ere used by the data separetor sub-sections. In the following discussions the term Reference Clock refers only to the signal in *Figure 7* that feeds the divide-by-2 end divide-by-N blocks. Also the term Divide-By-N counter is used for the counter driven by the Reference Clock, wherees the Progremmable Divider refers to the counter driven by the VCO.

In the DP8473, the Reference Clock of *Figure 7* is derived from a prescaler circuit that is operating at 24 MHz. The output of this prescaler circuit is 8 MHz for all data rates except 300 Kb/s. At 300 Kb/s the equivalent prescaler output is 9.6 MHz. The 24 MHz is intended to be a fixed frequency, but could be scaled lower for unique applications if desired.

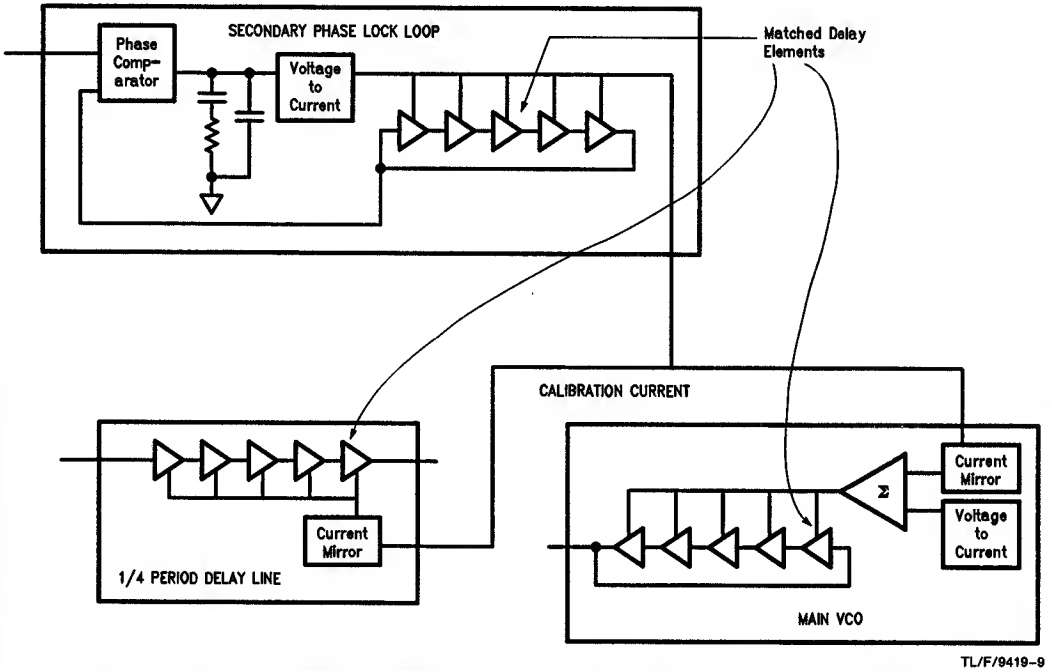
Under normal operation the Secondary PLL and the Primary VCO run at one half the Reference Clock frequency. Thus the Primary VCO's output is 4 MHz (except at 300 Kb/s where the VCO output is 4.8 MHz.) Operation at different data rates is accomplished by changing the Divide-By-N and Programmable Divider.

The basic operation of the Phase Locked Loop is fairly standard although there are several added features in the DP8473 PLL. The phase detector determines the phase



TL/F/9419–8

(a)



TL/F/9419–9

(b)

FIGURE 7. A more detailed Block Diagram of the DP8473 data separator, a) showing the zero phase start up, secondary PLL, control logic block and external R's and C's. b) Also showing a more detailed diagram of the secondary PLL, main VCO and delay line. This highlights the self calibration technique.

difference between two inputs. One input is always the divided output of the Programmable Divider.

The other input is either a reference frequency (derived from the Reference Clock) of twice the data rate while the data separator is in the idle mode, or when reading the disk the encoded data from the drive after it has passed through the quarter-period delay line.

While in the idle mode, the phase detector determines both phase and frequency information. This is accomplished by forcing the pulse gate to enable all phase comparisions. This state is set by the top two multiplexers of *Figure 7* which in idle mode select clocks generated by the Reference Clock to enable the pulse gate on every clock edge (hence all clocks are compared by the phase detector). By locking to the Reference Clock generated frequencies in both phase and frequency, the PLL is preventing from locking back to a harmonic of this frequency.

When the data separator is told to read the incoming data pulses, Read Gate is asserted. This changes the signals selected by the multiplexers. The top multiplexer switches to inputting the "raw" read data pulses into the pulse gate, and the bottom multiplexer sends read data delayed by a quarter bit period to the phase detector input. When this switch occurs, the Zero Phase Start-Up logic synchronizes the Programmable Divider's output to be in phase with the very next arriving data pulse. This causes the PLL to acquire lock to the data quicker since it is starting with a "near zero" phase error between the Programmable Divider and the encoded data.

The quarter-period delay line consists of a series of voltage controlled delay elements. The encoded data from the disk drive enters the beginning of the delay line. The output is derived from the output of one of the delay elements. The delay element used for the output depends upon the data rate used.

When locked to either the read data pulses or the reference, the phase detector issues either a pump up signal or a pump down signal depending upon whether the VCO should increase or decrease its frequency. The length of this pump signal is proportional to the amount of phase difference between the two input signals. When locked, the phase difference between the VCO and the delayed data will be small. In this case the width of the pump signal could be so small that the rise time may prevent the signal from ever being recognized by the charge pump. To ensure that the charge pump can recognize even the smallest pump signal, both pump up and pump down signals are asserted at each phase comparison and the appropriate signal is extended by an amount proportional to the phase difference between the two input signals. The pump signals are then subtracted from each other by the charge pump. Therefore, the rise time of the pump up or down signals will not degrade the performance of the charge pump.

The charge pump simply adds or removes an amount of charge proportional to the length of the pump signal to or from an external filter. The voltage of the external filter determines the frequency produced by the VCO.

Finally a synchronized clock and serial data signal is sent to the controller's deserializer by the Data Synch Logic Block. This circuit takes the output of the Programmable Divider and the Read Data pulses, and synchronizes these two signals, by centering the read data pulse in the appropriate Programmable Divider's clock cycle. This allows the controller to easily deserialize and decode the data pulses properly.

An additional block not shown here, but used on the DP8473 is the filter selection logic. This logic is used to select different filters for different data rates. The description and use of this circuitry is described in section 5.3.

3.2 Self Calibration

Normally, most VCO implementations would need an external precision capacitor (maybe trimmed) to set its center frequency. Also, the quarter period delay line would require an external trimming resistor to set the delay to exactly a quarter of the data rate. The actual delay of the delay elements used in these functions would normally vary from one part to another due to normal process variations. However, the DP8473 has been designed to eliminate the need for these external trims. There are actually two PLLs in the DP8473; the Primary PLL, and a Secondary PLL. The Primary PLL is used for data separation. The Secondary PLL is used to calibrate all of the delay elements used in the chip. This includes the quarter-period delay line and the main VCO.

The VCO of the Secondary PLL is a ring oscillator of delay elements. The amount of delay that each inverter produces is regulated by a control voltage which is the internally connected output of the Secondary PLL. The Secondary PLL is locked to the same frequency as the Primary VCO, half of the reference clock frequency. The delay elements used in the secondary VCO are identical to those used in the Primary VCO and are regulated by the same control voltage. There are also the same number of delay elements in each. Therefore, the center frequency of the Primary VCO is internally trimmed to exactly half of the reference frequency. Since the delay elements in the Secondary PLL have a known delay, any number of identical elements that are set by the calibration voltage will have a very accurate delay. Thus the quarter period delay line is just a chain of these delay elements that have the desired total length.

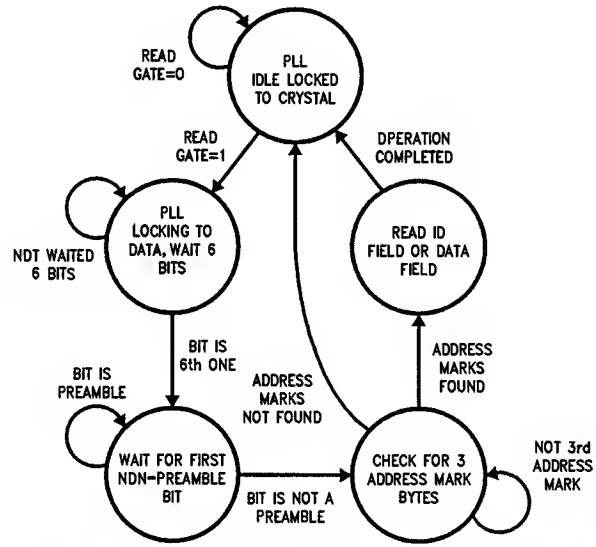
The delay elements used in the quarter-period delay line are also of the same type used in the secondary VCO. Because the delay of each element is accurately set by the Secondary PLL, there is no need for any trimmed tuning components for any of these circuits. Essentially, the only external passive components required for the DP8473 are for the filter(s) (two capacitors and a resistor per data rate), and a resistor to set the gain of the charge pump (ie. the amount of charge pump current).

3.3 Data Separator Read Algorithm

Since the DP8473 floppy disk controller incorporates the analog data separator, it takes advantage of close proximity of the controller and data separator blocks to implement a read algorithm that is much more sophisticated than previous floppy controller integrated circuits.

Before describing the details of the algorithm, a brief discussion of a disk read (or write) is necessary. When the controller is issued a read (or write) command, it is asked for a specified sector. The controller starts to look at the incoming MFM information. It scans this information, trying to locate the proper address field. In order to do this, the data separator is first told to lock to the disk data, and once locked the controller looks at the incoming information. In most controllers, the data separator is told to look at the data continuously until the correct sector address is found. This makes the data separator susceptible to being thrown out of lock, since the controller is not "watching" the data separator to see if it has maintained lock through the search process (which it can easily lose). Once the correct address is found the controller must then ensure that the associated data area is read, by re-locking to the incoming signal and then reading (or writing) the data.

Figure 8 shows in state diagram form the algorithm used by the DP8473 controller to ensure that disk data is correctly read (or written). This algorithm is much more sophisticated than previous generation controllers. The following describes the operation. When the controller is idle, the data separator is locked to the crystal/clock reference in a frequency comparison mode. (Frequency comparisons are made by disabling the pulse gate, and ensuring that all reference and VCO cycles are compared.)



TL/F/9419–10

FIGURE 8. State Diagram for a Data Separator Synchronization and Read Operation

When a read command is issued to the controller the controller asserts an Internal Read Gate signal to the data separator. This causes the PLL to switch from locking to the reference to locking to the data with the pulse gate enabled, and the primary VCO/divider started in phase with the next incoming pulse. The PLL waits 6-bit times to lock to the data. When the seventh bit arrives the data separator assumes that this bit is a preamble bit (thus an MFM clock bit). The controller/data separator then continues looking at the data until a non-preamble pattern is detected (ie. an MFM data pulse). It then checks to see if it has now encountered an address mark with the proper rule violation. If it has not, read gate is deasserted. The data separator returns to the idle state for 6 byte times, and then starts all over again.

If three address mark bytes are found then the data separator remains locked to the data while the controller looks to see if it has found the right address field. If the controller discovers that this field is not the correct address field then it deasserts read gate for 6 bytes, and tries again.

If the correct address field is encountered, the controller deasserts read gate during the gap between the address and data fields. It then re-asserts read gate, and follows the state diagram to read the data field (ie. looking for preamble, address marks etc.).

This comparison is done on a bit-by-bit basis, therefore ensuring that the PLL never tries to lock on an unwanted field for more than one bit time. In other words, the PLL will never loose lock. This algorithm provides a very fast lock to the data stream, and ensures that the data separator never falls out of lock while reading the data. Both of these features reduce the need to do retries of operations to ensure correct execution.

4.0 DESIGNING WITH THE DP8473 DATA SEPARATOR

The following section is a fairly in-depth description of the design characteristics of the PLL in the DP8473 controller. *(National Semiconductor cannot be responsible for the sanity of any one who ventures into this section. Hence we recommend using the filter values supplied in the datasheets.)*

Two elements determine the overall performance of a Phase Locked Loop: the loop gain and the loop filter design. When using the DP8473 both of these elements are controlled by the user. The amount of current in the charge pump circuit can be set with an external resistor. This will set the overall gain of the PLL. The filter is external to the DP8473 and is user definable. This gives the user the possibility of tailoring the data separator performance to his own application requirements and design criteria. The following information will present some tradeoffs that apply in choosing the external components for typical applications.

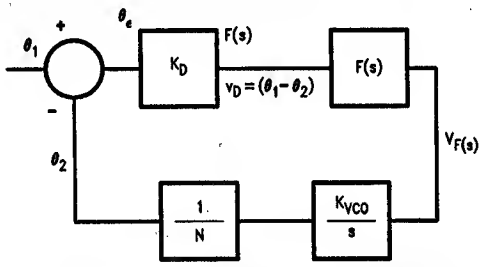
4.1 Basic Phase Lock Loop Theory

This section will first start with the basic control systems model for a second order PLL, and then apply these basic equations to the individual blocks that compose the data separator in the DP8473 Controller.

Initially Locked Model

In order to understand the behavior of the data separator and to discuss the tradeoffs of the different design parameters, some background in the theory of PLLs will be presented.

Figure 9 shows a diagram of a PLL which is assumed to be in a locked state. Each box contains the phase transfer function of the corresponding block and each node has the relative phase signal. The PLL is locked, which means that the VCO output and the input signal are at the same frequency and in-phase with each other (the phase error is a constant). This model is useful for understanding the phase locking process when the PLL is switched from the reference frequency to the incoming data.



TL/F/9419–11

FIGURE 9. The Block Diagram for an Initially Locked PLL Control System, Showing the Transfer Functions for Each Block

The frequency variations that this model take into account are assumed small enough so that the loop stays locked in frequency. Therefore, only the effect on the phase difference is considered. Also, it is easier to refer to bit shift tolerance in terms of phase. Hence the phase transfer functions yield the most appropriate information.

The concept of phase is very much related with the concept of time. The advantage of phase information is that it is independent of frequency of the signal and it is measured as a pure number (radians).

A simple RC filter model will be used to simplify the math. This is actually a good model because the effect of a second capacitor is only seen at high frequencies. This simplification allows the use of second order PLL theory that is easily available in literature. (See for example: R.E. Best, Phase Locked Loops, McGraw-Hill, 1984)

From *Figure 9* the closed loop phase transfer function for the loop can be derived using standard control system theory techniques, and reduces to:

$$H(s) = \frac{\theta_2(s)}{\theta_1(s)} = \frac{K_D K'_{VCO} F(s)}{s + K_D K'_{VCO} F(s)} \quad (1)$$

where $K'_{VCO} = K_{VCO}/N$, N being the number of VCO cycles between phase comparisons due to the divider that typically is inserted between the VCO and phase detector. The closed loop phase error function can be written as:

$$H_\epsilon(s) = \frac{\theta_\epsilon(s)}{\theta_1(s)} = 1 - H(s) =$$

$$\frac{\theta_1 - \theta_2}{\theta_1} = \frac{s}{s + K_D K'_{VCO} F(s)} \quad (2)$$

Now we'll evaluate the variables that appear in expressions (1) and (2).

The Charge Pump

In the phase detector/charge pump circuit a current is generated in the correct direction (positive or negative) every time the edges of the VCO/divider output and the incoming data pulses are not coincident. The current is a pulse with amplitude equal to I_{PUMP} and length equal to the phase error between the two signals. The pump current is zero for the rest of the period, so the average current is:

$$I_Z = \frac{I_{PUMP}\theta_\epsilon}{2\pi} \quad (3)$$

where θ_ϵ is the phase error between the VCO/divider and input pulses. The phase detector and charge pump gain is:

$$K_D = \frac{I_Z}{\theta_\epsilon} = \frac{I_{PUMP}}{2\pi} \quad (4)$$

where $I_{PUMP} = K_P \times I_R$. I_R is the current set by an external resistor at SETCUR pin. The current at this pin is: $I_R = V_{REF}/R_1$, $K_P = 2.5$, and $V_{REF} \cong 1.2V$. Thus combining equations:

$$I_{PUMP} = \frac{(2.5)(1.2)}{R_1} \quad (5)$$

Note that the maximum current that can flow to or from the charge pump is 500 μA, which corresponds to a resistor value of 3 kΩ. The minimum current is limited to 125 μA by stability and leakage constraints on the internal reference circuits. So R_1 must be smaller than 12 kΩ. In conclusion, the charge pump current and resistor can be set in the following range:

$$125\ \mu A \leq I_{PUMP} \leq 500\ \mu A$$
$$3\ k\Omega \leq R_1 \leq 12\ k\Omega$$

Any value in this range can be chosen, and usually the choice is dependent on the PLL filter capacitor's mechanical size and cost. We have chosen in the datasheet a value of 5.6 kΩ since it represents a good compromise of all these considerations.

The VCO and Programmable Divider

The VCO gain is defined as the ratio between a frequency change at the output vs. a voltage change at the input (FILTER pin). This value cannot be set by the user and has been designed to be immune to process, temperature and voltage variation. There is a variation of less than ±20% between different parts, and the typical value of K_{VCO} is:

$$K_{VCO} = 25\frac{Mrad/sec}{volt} \quad (6)$$

The actual value K'_{VCO} for expressions (1) and (2) differs by a factor of N. N is the ratio between the frequency of the internal VCO and the "instantaneous" frequency of the data. This takes into account both the factor due to the way the data is encoded, and the factor due to the internal programmable divider used for the data rate selection. The following table gives the value of N for different codes, data rates, and data patterns. K'_{VCO} can be derived from these values of N.

TABLE I. VCO Gain Reduction Factor for the DP8473 with a 24 MHz Crystal/Clock

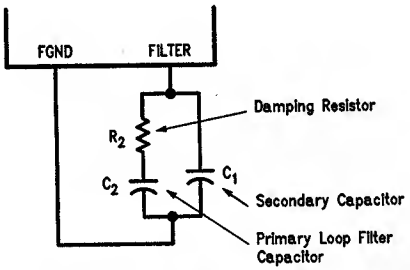
Data Rate	Code	Data Patterns	N
1 Mb/s	MFM	all 0's, all 1's	4
		010101 . . .	8
500 Kb/s	MFM	all 0's, all 1's	8
		010101 . . .	16
	FM	all 0's	8
		all 1's	4
300 Kb/s	MFM	all 0's, all 1's	16
		010101 . . .	32
250 Kb/s	MFM	all 0's, all 1's	16
		010101 . . .	32
	FM	all 0's	16
		all 1's	8
125 Kb/s	FM	all 0's	32
		all 1's	16

So for a 250 Kb/s MFM data rate N=16 and K'_{VCO} = 1.56 Mrad/set/volt.

The loop filter calculation is made assuming lock and acquisition during a preamble (all 0's pattern), so these values of N are used in the bandwidth and damping calculations shown later.

The PLL Loop Filter

Inside the data separator, the charge pump output is connected directly to the VCO input. A filter is attached externally to this point. The typical configuration of this filter is shown in *Figure 10*. The output of the phase detector/charge pump circuit is basically a current generator with a very high output impedance (hundreds of kΩ). This high impedance combined with the external capacitor, C_2, of the filter provide a small (close to 0) steady phase error after a frequency step in the input signal. The charge pump setting along with C_2 sets the bandwidth on the PLL. The DP8473's charge pump circuit eliminates the need for an external active filter. The resistor R_2 is the damping resistor and it controls the stability of the loop.



TL/F/9419–12

FIGURE 10. Simple Schematic of Typical DP8473 Data Separator Filter Configuration

The filter design is usually improved by adding another capacitor in parallel, C_1. This second capacitor is intended to improve the low-pass filtering action of the PLL. In our subsequent filter discussions, C_1 is ignored initially since its value will be much smaller than C_2.

In the DP8473, the input of the filter is a current from the phase detector/charge pump, the output is a voltage to the VCO. Therefore, the transfer function of the filter of *Figure 9* is simply its impedance:

$$F(s) = Z(s) = \frac{1 + sR_2C_2}{sC_2} \tag{7}$$

(As mentioned, we are ignoring the effect of C_1 for now.) Substituting these equations into (1) and (2) produces:

$$H(s) = \frac{\dfrac{K'_{VCO}K_D(sR_2C_2 + 1)}{C_2}}{\left(s^2 + s(K'_{VCO}R_2K_D) + \dfrac{K'_{VCO}K_D}{C_2}\right)} \tag{8}$$

This then reduces to a standard second order equation of the form:

$$H(s) = \frac{(2s\zeta\omega_n + \omega_n^2)}{(s^2 + 2\zeta\omega_n s + \omega_n^2)}$$

and similarly the error function has the form:

$$H_\epsilon(s) = \frac{s^2}{(s^2 + 2\zeta\omega_n s + \omega_n^2)} \tag{9}$$

In this discussion we won't be making use of the error function equation, however it is the basis of much of the acquisition equations which are used to derive *Figures 11* and *12*. The complete analysis is beyond the scope of this paper, but is discussed in detail by most of the references. From equation 8, and the standard second order equation we can solve for the bandwidth, ω_n and damping, ζ. The natural frequency is:

$$\omega_n^2 = \frac{K'_{VCO}K_D}{C_2}$$

$$\omega_n = \sqrt{\frac{I_{PUMP}K'_{VCO}}{2\pi C_2}} \tag{10}$$

By combining K_{VCO} and I_{PUMP} constants, the design equations for the PLL can be simplified, by introducing K_{PLL}, which is defined as the product of $V_{REF} \times K_P \times K_{VCO}$. By using the K_{PLL} constant the above equation becomes:

$$\omega_n = \sqrt{\frac{K_{PLL}}{2\pi N R_1 C_2}} \tag{11}$$

where N is the VCO divider as defined in Table I. The damping factor is given by:

$$2\zeta\omega_n = K'_{VCO}R_2K_D\frac{K'_{VCO}R_2K_DC_2}{C_2}$$

$$2\zeta\omega_n = \omega_n^2R_2C_2$$

$$\zeta = \frac{\omega_nR_2C_2}{2} \tag{12}$$

These two parameters (equations 11 and 12) will allow us to calculate the PLL filter components based on bandwidth, and damping. The closed loop phase transfer function shows that the PLL behaves like a low-pass filter. It passes signals for input phase signals whose frequency spectrum is between 0 and ω_n. This means that a second-order PLL is able to track a phase and frequency modulation of the input signal up to a frequency equal to $\omega_n/2\pi$, and it will not follow input variations of higher frequencies.

4.2 System Performance and Filter Design

The system performance of a data separator can be described by three main criteria.

1) Acquisition Time—ability to guarantee lock during a preamble.
2) Window Margin—ability to recognize data shifted in time from its ideal position (bit jitter) without incorrectly decoding it.
3) Tracking of Disk Data—ability to follow slow (<1 kHz) disk data speed variations.

The filter design must meet the requirements set by these performance characteristics. The two conflicting requirements are that the bandwidth must be large enough to ensure proper locking to the data stream, but as small as possible while tracking the data to maximize bit shift rejection and window margin. Primarily the filter sets the bandwidth, and it is determined by the required acquisition time as shown later.

To illustrate this, a numerical example (for 500 Kb/s data rate) is presented following the design considerations step by step. After we have completed the paper design a discussion of performace measurements is provided. Once the initially calculated paper values are chosen, then real measurements must be made, and adjustments to these values are decided upon.

Acquisition to the Data Stream

Acquisition means to achieve phase lock and to bring the phase error of the VCO to zero, or close to it. This includes acquisition of phase lock to either the data input or to the reference frequency. The lock mechanisms for these two cases may be different.

At the moment just before Read Gate is asserted, it will be assumed that the VCO is locked to the reference frequency. It has been locked for a relatively long period of time, therefore the phase error between the VCO output and the reference frequency is nearly zero. Because the system is initially locked, the initially locked model can be used.

When Read Gate is asserted by the controller, the input to the phase detector is switched from the reference frequency to the data stream. This is an instantaneous change of both phase and frequency to the input of the phase detector. The loop must be designed to assure that it can achieve both phase and frequency lock to the incoming data. Phase and Frequency Lock implies that the steady state phase and frequency error at the phase detector input is near zero.

The goal is to lock to the data stream within the length of the preamble, very often half of the preamble to increase the probability of locking successfully. In fact, during a preamble the data pulses are relatively free of bit shift and the frequency is constant. There are two basic requirements to ensure that the data separator correctly locks to the data stream in the required amount of time.

1. The loop bandwidth must be large enough to ensure that phase and frequency error of the VCO goes to zero within the required time (usually within ½ the length of the preamble). This implies that the shorter the preamble the larger ω_n.
2. The filter must also be designed to guarantee that a data pulse will never fall out of the data window during the lock process. The peak phase error during acquisition must be less than ½ of a data or clock window (i.e. or $< \pi/2$). If the filter is incorrectly designed and the data pulse falls outside the window (called cycle slipping) during acquisition, the loop may never lock within the desired acquisition time and the encoded data will not be decoded correctly. This requires that to guarantee lock over a wider variation of data rate, a larger ω_n is required.

Both of these requirements can be approximately derived from *Figures 11* and *12*. These curves plot relative phase error normalized to ω_n versus time in units normalized to ω_n. This period of time, and the amount of phase error present is dependent upon ω_n and damping.

For the first requirement, the phase error settles close to 0 in about:

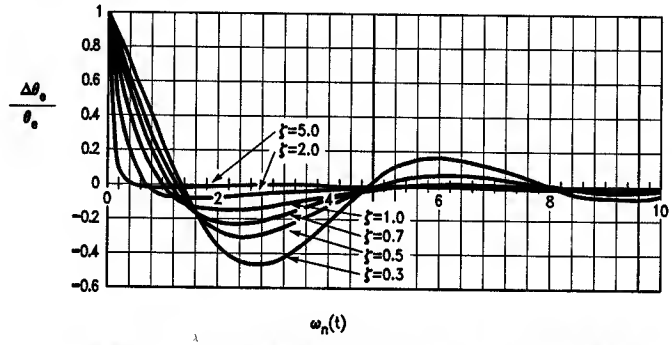
$$t_{acq} = \frac{5}{\omega_n} \tag{13}$$

This equation yields a minimum starting bandwidth, and is valid for systems where the speed variation of the incoming data is small (±1–2%).

For the second requirement, the peak phase error, $\theta_{e(PEEK)}$, during acquisition can be determined from *Figures 11* and *12*. The design must ensure that the sum of this peak phase error due to a phase step, $\theta_{e(PHASE)}$, the peak phase error due to a frequency step, $\theta_{e(FREQ)}$, and the phase error due to PLL noise and non-linearities, $\theta_{e(PLL)}$, must all be less than $\pi/2$. In equation form:

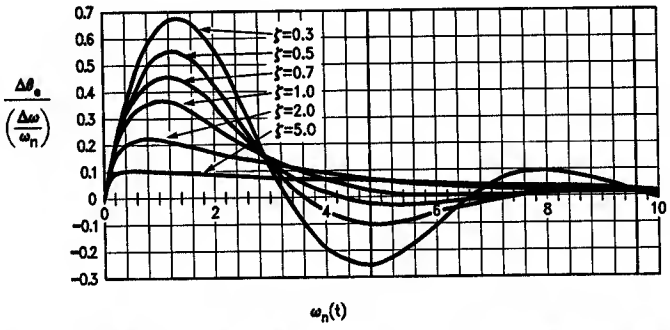
$$\theta_{e(PEAK)} = \theta_{e(FREQ)} + \theta_{e(PHASE)} + \theta_{e(PLL)} < \frac{\pi}{2} \tag{14}$$

Thus the sum of the peak phase errors for a phase step and a frequency step must be calculated.



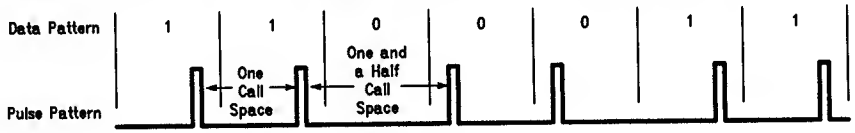
TL/F/9419–13

FIGURE 11. A Plot of Normalized Phase Error of a PLL to a Phase Step Input



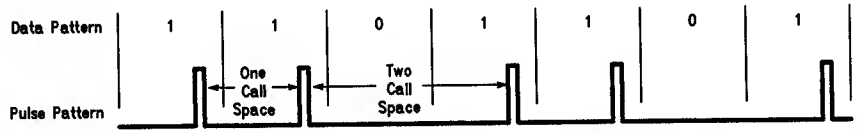
TL/F/9419–14

FIGURE 12. A Plot of Normalized Phase Error of a PLL to a Frequency Step



TL/F/9419–15

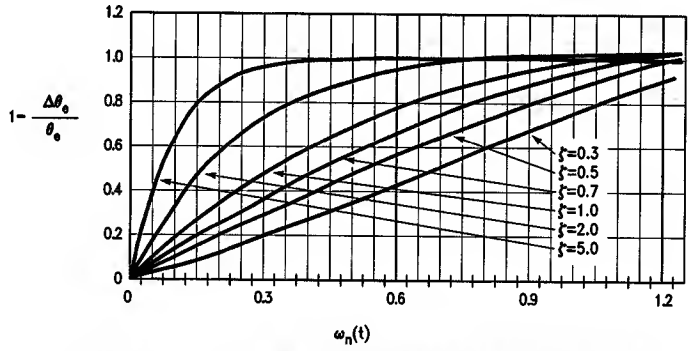
(a)



TL/F/9419–16

(b)

FIGURE 13. Two "Worst Case" Data Field Patterns for Measurement of PLL Bit Shift Tolerance a) "11000" Pattern with 2/3 μs Pulse Spacing (at 500 Kb/s), and b) A "DB6" ("110") Pattern with 2/4 μs Pulse Spacing (at 500 Kb/s)



TL/F/9419–17

FIGURE 14. A Plot of Normalized Bit Shift Resistance Versus Time for a PLL

For determining the peak phase step error, the value of $\theta_{e(PHASE)}$ is the maximum Y-axis value of the chosen curve multiplied by the input phase step, and the result is in radians. For example with a damping of 0.7 a maximum error of −0.20 occurs at about 3 ω_n. If the maximum phase step is $\pi/2$ when switching to the data, then the peak phase error is 0.1 π radians. The choice of which curve to use depends on damping factor desired.

Since a frequency step is also present at the transition of READ GATE, the peak phase error of the frequency step given by the normalized plot in *Figure 12* must also be derived. The phase error can be calculated by reading the peak Y-axis value from the desired curve and using:

$$\theta_{e(FREQ)} = Y\frac{\Delta\omega}{\omega_n} \qquad (15)$$

to determine $\theta_{e(FREQ)}$. Where Y is the value read from the Y axis, and $\Delta\omega$ is the maximum frequency step times 2π. The maximum frequency step is the worst case frequency variation of the data being read from the disk drive, which is the sum of the MSV and the ISV.

With the equations for loop bandwidth, damping factor, and the relationship between acquisition time and bandwidth, the following example demonstrates the first steps at arriving at the loop filter components.

Example 1: Design a data separator using the DP8473. Determine the loop bandwidth, dampening factor, and C_2, R_1, R_2 component values for a data separator that decodes MFM data at a data rate of 500 kbits/sec. The preamble is 12 bytes long, and the total MSV/ISV is ±6%.

Select a value of pump current resistor. For example 5.6 kΩ.

Find out the minimum acquisition time required. Generally, half preamble which is 6 bytes. Thus:

$$t_{acq} = ((6 \times 8)\text{ bits}) \times 2\ \mu\text{s/bit} = 96\ \mu\text{s}$$

Next calculate ω_n based on the larger of the two acquisition requirements. The first requirement for ω_n is: $5/\omega_n < t_{acq}$; Thus:

$$\omega_n > 52.5\ \text{Krad/sec.}$$

Calculate ω_n for the second acquisition requirement, i.e. ensuring the maximum phase error is less than $\pi/2$. Due to the Zero Phase start-up block within the Data Separator, the maximum phase step when switching to the data is $\pi/8$. We'll choose a damping of 0.7. From *Figure 11* the maximum overshoot is 0.2 ($\pi/8$) = 0.08 rad (Note 1.0 rad = 314 ns at 500 kb/s). Assume that the data separator contributes a total 0.1 rad noise error. This is for charge pump asymmetry, delay line variation, and VCO jitter.

Using equation 18:

$\theta_{e(PEAK)} = \theta_{e(FREQ)} + 0.08$ rad + 0.1 read < $\pi/2$ or . . .

500 ns > 25 ns + 31 ns + $\theta_{e(FREQ)}$ which results in:

$\theta_{e(FREQ)} < 443$ ns or 1.41 rad

This yields the maximum tolerable phase error due to a frequency step (which is dependent on ω_n).

5

First find the maximum frequency step in radians that the data separator must undergo. The design requirement was 6%, but in order to account for gain variations in the data separator some margin on top of this is required, so we will design to 8% total speed variation. Thus:

$$\Delta\omega = 0.08\,(500k)\,2\pi = 251\ \text{Krad}$$

The relative phase step error from *Figure 12* using a damping factor of 0.7, is Y = 0.45, plugging into equation 15:

$$\omega_n \geq \frac{\Delta\omega}{\theta_e} = \frac{0.45\,(251\ \text{Krad})}{1.41} = 80\ \text{Krad/s}.$$

The larger of the two calculated ω_n's is 80 Krad/sec, so that is the chosen bandwidth.

$$C_2 = \frac{K_{PLL}}{2\pi R_1 N \omega_n^2} \qquad (16)$$

$$= \frac{75\ \text{Mrad}}{2\pi(8)(5.6\ \text{k}\Omega)(80\ \text{k}^2)} \approx 0.041\ \mu\text{F}$$

We will round down to the next lowest standard value of 0.039 μF.

$$R_2 = \frac{2\zeta}{\omega_n C_2} \qquad (17)$$

$$R_2 = \frac{2\zeta}{C_2\omega_n} \approx \frac{2(0.7)}{(0.039)\,(80K)} \approx 450\Omega$$

In the above example we have calculated a set of component values for the acceptable bandwidth based on acquisition. This calculation yields a good starting point from which experimental measurements can be made, but may not be the optimum values. Depending on other considerations we may decide to chose a value of ω_n that is slightly different depending on window margin, or bit shift performance; as will be shown.

Theoretical Dynamic Window Margin Determination

Previously in section 2.4 Window Margin was discussed in terms of distortions of the data that degrade the window margin. Here, using a model for these distortions we will arrive at a calculation of the expected Dynamic Window Margin for the DP8473 analog PLL. The effects of Window error, VCO jitter, and PLL response to a previous or present bit shift all cause a reduction of the available window margin available. (Remember the goal is to maximize the total available bit window.) The following analysis provides a feel for the amount of degradation due to various parameters, and serves to provide an indicator of the expected PLL performance. The following is a list of parameters that cause loss of window:

Internal Window Error (or static phase error): The window error is related to the accuracy of the internal delay line in the data path before the phase detector. As explained previously, this delay line is automatically trimmed using the crystal frequency as reference. The static phase error is the sum of two factors. One of these factors is the difference between the data stream frequency and the nominal and unavoidable internal mismatches. Another contributing factor is charge pump leakage. This factor causes a perceived phase error that is equivalent to varying the delay line length. This is usually ≥ 2%–4%.

VCO Jitter: The VCO jitter is caused by the modulation of the VCO frequency with secondary VCO frequency, crystal oscillator and other noise. This can account for another 2–5% percent of error.

PLL Response: The PLL response to a data bit shifted from its nominal position because of noise or jitter is directly translated in a margin loss for the bits following any shifted bit. For a highly accurate PLL circuit this is the primary source of error, and it typically results in a window loss of up to 20%, depending on data pattern and frequency variation constraints.

All of these degradations are summed into the Window Margin specification.

$$\theta_{wm} = (\tfrac{1}{2}\ \text{Bit Window}) - \theta_{e(PLL)} - \theta_{e(SWL)} \qquad (18)$$

This yields the margin loss, where θ_{wm} is the total window margin or the total amount of a half bit cell in which a data pulse will be properly recognized, $\theta_{e(PLL)}$ is the error due to PLL response, and $\theta_{e(SWL)}$ is the total error contributed by imperfections of the PLL circuitry (including delay line accuracy, leakage, and noise).

The window margin loss contributed by the device accuracies are relatively straightforward, and are supplied by National. The more difficult task is to determine the window margin loss due to the PLL response.

Tracking of the Disk Data

The bit shift produced by an average disk depends on the pattern of encoded pulses recorded on the media. Pulses that are placed close together appear to push each other apart when they are read back. This is the primary cause of bit shift.

The PLL tolerance to bit shift is dependent on the amount of data bits that are shifted, and the data pattern. It can tolerate more bit shift by individual bits if only a few of the bits within a pulse stream are shifted. However, if most of the data read by the PLL is shifted (both early and late), the loop is constantly correcting itself and is never really phase locked. Because it is not phase locked the maximum tolerable bit shift is less.

Some data patterns are better at determining PLL performance than others. These are patterns that are particularly difficult for a PLL remain locked to while the data pulses are shifted early and late. For example a bit pattern of "11000" is difficult to decode since it has pulses alternately spaced by 1 and 1.5 bit cells. See *Figure 13a*. This is difficult to decode in some cases because under maximum bit shift conditions all pulses are equally spaced.

Another bit pattern that is difficult to decode is a repeated bit pattern triplet of "110" (or "101" also referred to as a "DB6" pattern), which has a pair of pulses one bit cell apart, and the next pair two bit cells apart. This pattern is particularly difficult to decode because it contains two pulses of minimum spacing, followed by two maximum spaced pulses. This pulse pattern is shown in *Figure 13b*.

Unlike the acquisition process described earlier, the PLL must largely ignore individual bit shifts during the tracking phase. The PLL should only follow the longer term average

data rate. The desired PLL response during the tracking phase is somewhat different from the response required during the acquisition phase. Instead of a high bandwidth to decrease the lock time, a low bandwidth is preferred to prevent the PLL from following individual bit shifts. When choosing the filter bandwidth, the lowest possible value should be used that still satisfies the acquisition time requirement.

Figure 14 can be used to determine the theoretical window margin. This curve of phase bit shift resistance plots the amount of error introduced in the PLL's VCO by a phase error. The following example will show the steps involved in calculating expected window margin, and illustrate some concepts of tracking data.

Example 2: Determine the total dynamic window margin for a loop with a bandwidth of 90 Krad/sec, a damping of 0.7, and at a 500 Kb/s data rate. The intrinsic circuit errors amount to 0.1 radians of the window. Calculate the margin for both the 110 and 11000 pattern.

First step is to calculate the bandwidth of the PLL while tracking these data patterns. The bandwidth is the square root of the ratio of the pulses per bit cell relative to preamble data, multiplied by the bandwidth. Preamble data has 1 pulse/bit cell, and a 110 pattern has 2 pulses per 3 bit cells, while a 1100 pattern has 4 pulses for every 5 cells. Thus:

$$\omega_{n(110)} = (90\ \text{Krad/sec}) \times \sqrt{0.66}$$
$$= 72\ \text{Krad/sec}$$
$$\omega_{n(11000)} = (90\ \text{Krad/sec}) \times \sqrt{0.8}$$
$$= 81\ \text{Krad/sec}$$

To analyze the problem, we need to look at two bits. The present bit which has an early phase step, and the next bit which has an equal late phase step. We must determine how large a shift in the first bit can be tolerated such that the same amount of shift in the second bit will still fall within the proper window. In equation form:

$$\theta_{e(2)} + K_{WM}\Delta\theta_{e(1)} + \theta_{PLL} \geq \theta_T \qquad (19)$$

where $\Delta\theta_{e(1)}$, the shift of the first bit, is multiplied by K_{WM}, which is the affect the first bit has on the VCO when then next bit arrives. θ_T is the total window available (in this case ±500 ns). θ_{PLL} is the static error degradation due to the DP847x and is 10% of 500 ns or 50 ns. $\Delta\theta_{e(2)}$ is the maximum phase error of the second bit. We can solve for $\Delta\theta_{e(2)}$ assuming that $\Delta\theta_{e(1)} = \Delta\theta_{e(2)}$:

$$\Delta\theta_{e(2)} = \frac{\theta_T - \theta_{PLL}}{1} + K_{WM} \qquad (20)$$

The value of K_{WM} is the value read off the y-axis of *Figure 14*, at a time normalized to ω_n. This time is the time between two bits or:

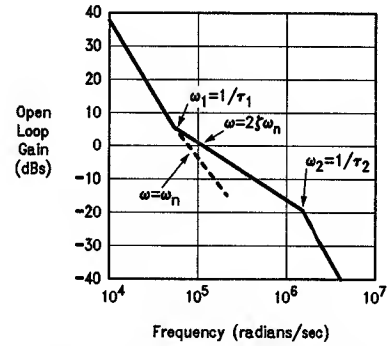
For a "110" pattern $\omega_n(t)$ = (4 μs) (72 Krad/s) = 0.29, resulting in a value from *Figure 14* of K_{WM} = 0.37. And for a "11000" pattern $w_n(t)$ = (3 μs) (81 Krad/s) = 0.24, which results in K_{WM} = 0.28. Using 0.37, the window margin is:

$$\Delta\theta_{e(2)} = \frac{500\ \text{ns} - 31\ \text{ns}}{1.37} = 342\ \text{ns}$$

In terms of percentage this is 100*(342/500)% = 68%. Note that this is the window margin at nominal frequency.

Open Loop Bode Plots and the Second Capacitor

Figure 15 shows the open loop gain Bode plot for the second order PLL. This plot is useful as a double check to make sure that we have a stable design and is important to show the affect of the second capacitor in the filter. In this plot, it is assumed that the charge pump output impedance is infinity.



TL/F/9419–19

FIGURE 15. Bode Plot of Open Loop Gain of DP8472/3/4 Using a Typical Filter at 500 Kb/s (from Example 1)

As can be seen, the gain starts off with a slope of 40 dB per decade due to the two poles of the filter and VCO. The phase angle starts at −180°C. The stabilizing zero is introduced at $\omega_1 = 1/R_2C_2$, and causes the slope to change to 20 dB per decade. ω_n is the extrapolation of 40 dB/dec line to 0 gain, and the actual crossing point is $\omega = 2\zeta\omega_n$. Example 4 discusses how to plot this curve.

The further reduce the effect of unwanted changes in the VCO phase the second capacitor can be added to the filter. This capacitor, C_1, introduces a pole, *Figure 15*, between the loop natural frequency and the data rate frequency. The pole due to the second capacitor occurs at $\omega_2 = 1/\tau_2 = 1/R_2C_1$. This capacitor provides further attenuation of bit shift caused frequency components, and the pump pulse noise, both of which have frequency components that are around the data frequency. The only considerations in choosing the value of this capacitor are related to the stability of the loop, and inadvertently affecting ω_n. A good criterion for stability is that the Bode plot of the open loop gain, *Figure 15* must cross the 0 dB gain with a slope of 20 dB/dec, ie. before the break caused by C_1's pole.

To determine a simple method for deriving C_1, we must look at the open loop gain of the PLL along with the transfer function of the loop filter. The open loop gain is:

$$\frac{\theta_2}{\theta_1} = \frac{K_D K'_{VCO}}{s} F(s) \tag{21}$$

Where F(s) is the filters transfer function:

$$F(s) = Z(s) = \frac{\left(\frac{1+sR_2C_2}{sC_1}\right)}{\left(\frac{sR_2C_1C_2}{(C_1+C_2)}\right)} \tag{22}$$

Combining these two equations and manipulating we find that the second pole occurs at:

$$\omega_P = \frac{1}{R_2C_1} \text{ (confirming } \textit{Figure 15}\text{)} \tag{23}$$

This is assuming $C_2 >> C_1$ (which we ought to assume to maintain the validity of previous filter assumptions).

The zero introduced by C_2 and R_2 should be designed to be close to the 0 dB gain crossing. Its frequency is $\omega_z = 1/R_2C_2$. The frequency of the pole due to R_2 and C_1 is approximately $\omega_P = 1/R_2C_1$. This pole must not significantly change the slope around the 0 dB line. If we choose $C_1 = C_2/20$ the effect on the slope of the transfer function is less than 1 dB/decade at the frequencies around the 0 dB gain line crossing. Thus as a guide:

$$C_1 \leq \frac{C_2}{20} \tag{24}$$

Example 3: For example 1, determine C_1.

Very simply for example 1a:

$$C_1 \leq \frac{0.039\ \mu F}{20} \cong 2000\ pF$$

The 1/20 factor provides the approximate value for C_1.

Example 4: Plot the Bode diagram of the open loop gain for the DP8472 with $C_2 = 0.027$, $C_1 = 1000$ pF, $R_1 = 5.6$ kΩ, $R_2 = 545\Omega$.

There are two easy methods of doing this. One method involves determining the open loop gain at a low frequency, where the poles and zeros don't have any affect, and then using this gain point at a start drawing the properly sloped lines to the break point frequencies. A second method is to calculate the ω_n, and $2\zeta\omega_n$ frequencies.

For the first method, first calculate the locations of τ_1 and τ_2 in *Figure 15*. Thus

$$\omega_1 = \frac{1}{\tau_1} = \frac{1}{R_2C_2} = \frac{1}{(545\Omega)(0.027\ \mu F)} = 6.8 \times 10^4$$

$$\omega_2 = \frac{1}{\tau_2} = \frac{1}{R_2C_1} = \frac{1}{(545\Omega)(1000\ pF)} = 1.8 \times 10^6$$

Now pick a point that is below ω_1, and calculate the open loop gain, which is:

$$K_{LOOP} = 20 \log \left[\left(\frac{K_{VCO}K_PV_{REF}}{2\pi R_1 N\omega} \right) \left(\frac{1}{\omega C_2} \right) \right]$$

At $\omega = 10^4$, the K_{LOOP} is:

$$K_{LOOP} = 20 \log \left(\frac{12 \times 10^6}{R_1C_2\omega^2 N} \right) = 39\ dB$$

Now draw a line from $\omega = 10^4$ to ω_1 with a 40 dB/decade slope. Then at ω_1 draw a line to ω_2 with a 20 dB/decade slope, and finally draw a line from ω_2 with a 40 dB/decade slope.

To understand the affect of C_1, the additional attenuation introduced can be determined as using:

$$A_P(w) = \frac{1}{1 + \frac{\omega}{\omega_p}} \tag{25}$$

Using our previous example 1:

$$\omega_P = \frac{1}{1000\ pF\ 545\Omega} = 1834\ Krad, \text{ and}$$

$$\omega = 2\pi\ (500\ kHz) = 3142\ Krad/sec.$$

Thus

$$A_P(w) = \frac{1}{1 + \frac{3142}{1834}} = 0.37$$

which yields an additional 9 dB of attenuation.

Choosing Component Tolerances and Types

One of the most often asked questions is how accurate should the filter and charge pump resistors and capacitors be? The answer depends on how accurate a data separator is required. For a good performance design, the following criteria can be followed for each component:

R_1: The pump set resistor's tolerance affects the loop bandwidth, ω_n. The loop bandwidth directly affects window margin. Due to the square root relationship, a 5% change in this resistor changes ω_n by 2.5% which in turn affects the window margin by 1–2%. It is thus recommended that R_1 be a 1% resistor. A standard carbon or metal film resistor with a low series inductance should be chosen.

C_2: The main filter capacitor also affects ω_n in the same way as R_1 so it too should be relatively accurate. 5% is recommended. This capacitor should be a very good quality capacitor, with good high frequency response and low dielectric absorption. Mica is a good choice although maybe too expensive. Polypropalene and metal film are good as well. Avoid Mylar or Polystyrene.

R_2: This resistor has a much lower affect on window margin, and thus standard 5% resistors can be used.

C_1: The second capacitor's accuracy in not critical 10%–20%, but its high frequency characteristics should be quite good, similar to a good high frequency power supply decoupling capacitor.

5.0 ADVANCED TOPICS

The following sections discuss several specialized areas of evaluation and design of the PLL for the DP8473 controller. Also a short discussion of the crystal oscillator design considerations is given.

5.1 Design and Performance Testing

Testing data separators can get rather complicated. Once the PLL circuitry, gain, bandwidth, and damping are set, then data separator lock range testing, window margin testing, and finaly bit error rate testing may be undertaken. This testing can require some fairly sophisticated setups, and is time consuming. To help the designer get started, a rigorous approach to floppy disk PLL design verification is described with reference to desired performance and available equipment. If the designer is not concerned about optimum performance for custom applications, then the values for the PLL filter and pump resistor provided in the DP8473 datasheet should prove more than adequate.

Step 1—Calculating the Filter/Pump Resistor: Following the examples above, the optimum "paper design" filter components should be calculated (or use the values provided in the datasheets and tweak them).

Step 2—Testing Lock Range and Damping: For characterization of the PLL's acquisition, a fairly simple setup can be used, which utilizes a pulse generator to provide a pulse train that simulates a preamble to be input into the data separator's read data input. A second synchronous square-wave that is 20–50 times the period of the data pulses is applied to read gate. The frequency of the read data pulses should be varied from the nominal data rate to the limits of the desired lock range, and the lock range requirement should be verified by monitoring the Filter pin, and the Pump outputs. *Figure 16* shows a typical setup, and some typical waveforms on the pump and filter pins during acquisition. *Figure 16b* shows a proper locking PLL which does not exhibit "cycle slipping". Cycle slipping is denoted by the saw tooth waveform on the filter pin, which can be seen by the locking shown in *Figure 16c*.

In both *Figure 16b* and *c* the total amplitude, ΔV, of the filter pin waveform is typically less than 200 mV.

The object is to adjust the PLL bandwidth to ensure that when locking over the desired range of data rates (for example 500 Kb/s ±6%) that no cycle slipping occurs. If slipping does occur then the bandwidth should be increased.

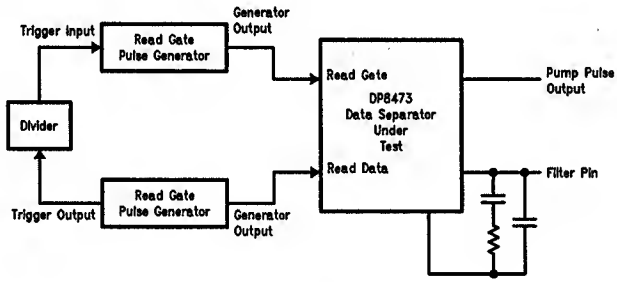
A second piece of information that these curves provide is verification of the damping of the PLL. As in *Figure 16b* the filter pin waveform should slightly overshoot, and then (maybe) slightly undershoot the eventual locked voltage. If there is very little overshoot then the loop may be overdamped, and if the filter pin voltage "rings" for a few cycles the loop is probably underdamped. For example, *Figure 16c* not only cycle slips, but does not overshoot, therefore the loop bandwidth may be fine, and the loop is just too heavily damped.

Step 3—Window Margin Evaluation: Usually to perform this test a disk simulator should be used to simulate the worst case drive read data conditions, and measure the error performance of the data separator. This simulator can be used to vary the following parameters:

1. Motor Speed Variation
2. Instantaneous Speed Variation
3. Instantaneous bit shift
   (to determine the window edge)
4. Data pattern.

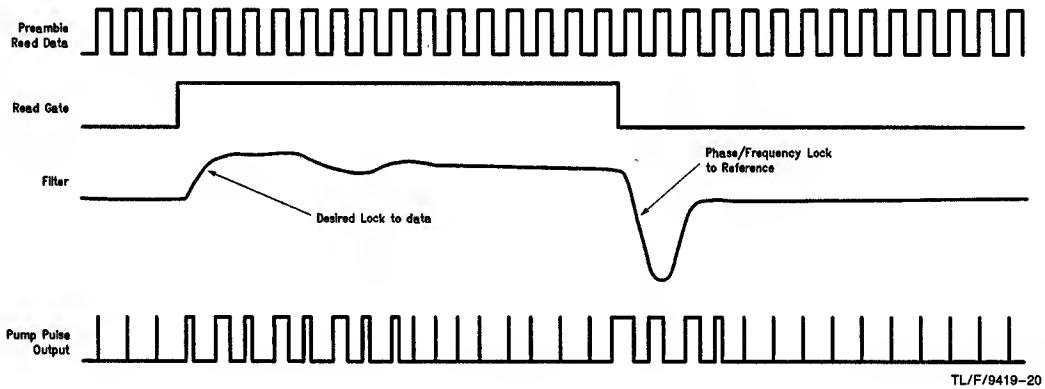
The test setup shown in *Figure 17* can accomplish some of this testing. The disk simulator looks like a formatted disk to the controller/data separator. To test the data separator, software in the host computer performs a repeated series of read operations over a period of time, while the designer programs the disk simulator to vary the data rate from one end of the lock range to the other. At each data rate the bit shift is increased until the error rate increases above a minimal threshold.

This testing should measure window margin over the entire lock range, and under conditions as described in section 2.4. Typically this process is a trial-and-error process. The bandwidth and damping can be adjusted based on the results of these tests.



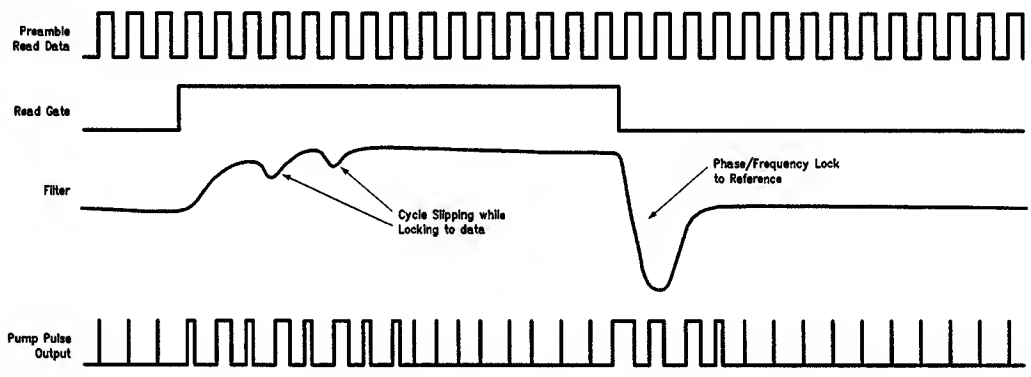
TL/F/9419–19

(a)



TL/F/9419–20

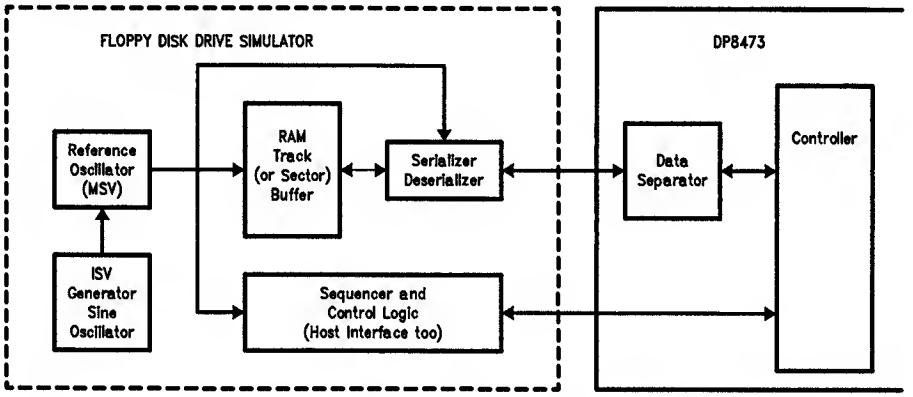
(b)



TL/F/9419–21

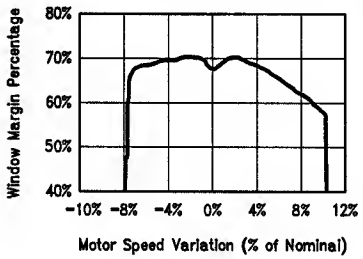
(c)

FIGURE 16. Simple PLL lock/performance testing; a) Typical frequency generation hardware to generate read gate and preamble for various frequencies. b) Using this hardware a typical lock acquisition showing key signals. This is a proper lock waveform. c) This shows an unreliable lock to a frequency beyond the lock range of the PLL. Cycle slipping occurs because the PLL is unable to respond quickly enough to the frequency step.



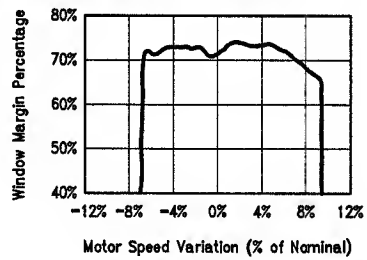
TL/F/9419–22

FIGURE 17. Block Diagram of Connection of Disk Simulator to Data Separator to be Teated



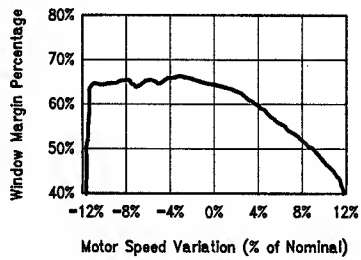
TL/F/9419–23

(a)



TL/F/9419–24

(b)



TL/F/9419–25

(c)

FIGURE 18. Various window margin graphs. a) Data separator with slightly long delay line and fairly normal lock range; b) data separator with short delay line; c) a relatively poor data separator with a long delay line, and wide bandwidth.

ISV may also be simulated. Unfortunately most disk simulators do not easily test for this. Therefore it is very likely that the window margin measurement will be made with only MSV variation initially. This may also be desirable since MSV-only testing will yield a better understanding of the PLL's lock range. If MSV-only testing is done initially then the design of the PLL and total frequency range for evaluation should be the sum of the desired MSV and ISV. For example, a design requirement of ±3% ISV and ±3% MSV can be approximated by ±6% MSV.

If an MSV-only test is done, it may be useful to then follow this with a simple ISV test just to double check that the loop will follow the ISV properly with little degradation in window margin. It is also sometimes useful to vary the ISV frequency until the window margin degrades, just to give an indication of how high the ISV frequency can go. A typical PLL's ISV performance should not degrade until the frequency is greater than about 800 Hz.

Step 4—Considering Temperature, Supply and Device Variations: The next step is to take the above "optimum" filter and simulate variations in gain induced by temperature, supply and processing. This is accomplished with the same disk simulator setup as in step 3. To simulate these variations, the designer need only vary the pump resistor's value. This will affect the open loop gain identically to device variations. The above tests should be re-run with a minimum and maximum resistor. If the performance "falls-off-a-cliff" then some compromises and adjustments to the filter may be required.

For relatively small temperature ranges a resistor variation of ±10–15% should be adequate. For full 0°–70° simulation, resistor variation of ±20% is a more accurate reflection of the DP8473 performance. If for some reason the performance of the data separator cannot be maintained at the desired window margin, then trimming the pump resistor may be needed to meet performance over the full process spread.

The final simulation involves varying the delay line length. It is possible to simulate variations in the delay line by forcing a leakage current onto the filter pin (at the VCO input) using a pull up or pull down resistor. This leakage current will cause a phase error within the loop and will cause the loop to act as if the delay line length were changed. Before actually running dynamic window margin tests, the designer must determine the actual length of the delay line, and then adjust this length to the limits specified in the datasheet. Then at each limit the dynamic window margin test can be performed.

In order to measure the length of the delay line a static window margin test can be performed. This test uses a disk simulator with a format that has all 0's or 1's data fields. Within the data field one bit is shifted until the PLL mis-decodes the data. Using the maximum tolerable bit shift number, the delay line length can be extrapolated. This same measurement can be done with various filter pull up/down resistors. By proper resistor selection the limits of the delay line length can be simulated. Using these simulated delay line techniques a dynamic window margin test can be run, and the resultant PLL performance can be characterized.

Step 5a—Bit Error Rate Measurements: This test is performed as a verification of the final total system. It consists of putting together a complete floppy drive, floppy media, separator, and controller system then running long term read/write tests randomly across the disk media. For a known number of read/writes and the resultant value of read errors, a number can be derived that is the ratio of bit errors to total number of bits read. This test proves the integrity of the entire system, and should be performed over some manufacturing spread of products. While this test is useful to verify the complete system integration, the data separator is only one small part of the total contribution to the total system error rate.

Step 5b—Real World Worst-Case Tests: As a final analysis and proof that the data separator is solid it is often useful to test the data separator on a known "worst case" drive and disk. Generally the evaluation is done with a disk that is recorded off speed and off track. This ensures that a maximum amount of bit shift and speed variation is present. The main problem is to ensure that the disk still has acceptable data. If excessive errors are encountered, evaluation of the types of error that are occurring can be used to determine whether the bandwidth of the PLL needs to be increased (acquisition related errors) or decreased (bit shift related errors).

Alternate Simple In-System Data Separator Evaluation

Provided here are some quick tips on evaluating the data separator without any test equipment, but just by running long term in-system tests.

1. If after some initial testing a lot of sector or ID address mark or data address mark not found errors are given by the controller, then in all likelihood the data separator is not locking properly. The bandwidth of the loop should be increased, or possibly the loop is too heavily overdamped.
2. If the disk controller is experiencing a large amount of address or data field CRC errors, then probably the loop is being sent out of lock by bit shift noise. Thus, the gain of the loop may be too high or the loop is underdamped.

5.2 Understanding the Window Margin Curves

Careful analysis of a window margin curve can yield quite a bit of information about the data separator characteristics. *Figures 18a*, *b*, and *c* show some typical window margin plots. These curves were made using a disk simulator that outputs a "reverse write precompensated" bit shift pattern to the data separator.

These curves plot the maximum bit shift tolerance (vertical axis) versus motor speed variation (MSV-only) (horizontal axis). The controller was programmed to perform a repetitive single sector read, while the simulator outputs a formatted track at the programmed MSV and bit shift amount. All the data read with MSV and bit shift amounts that fall under and within the curve (shaded area) can be consistently read correctly. All data read with MSV and bit shift amounts outside and above the curve either could not be correctly located by the controller, or had errors in it.

The first thing to note is that as the MSV variation from the nominal frequency increases, there is a point at which the PLL performance drops to zero bit shift (the vertical lines of the curve). This is an indication of the lock range of the PLL. Thus for *Figure 18a* the lock range is −8% to +10%. This asymmetry in the lock range is typical of the DP847x series PLLs and is due to a slight skew in the charge pump. This

skew adds some bit shift rejection improvement. *Figure 18b* has a lock range of −7% to +9%, while *Figure 18c* has a much wider lock range of nearly ±12%. This is an indication of the loop bandwidth. Here *Figure 18a* has the lowest bandwidth, then *Figure 18b*, and finally *Figure 18c* has the highest bandwidth.

Another characteristic of each of these curves is the downward slope in the positive MSV direction. The start of this slope is the point at which the delay line becomes longer than the ¼ period of the data rate. For example, in *Figure 18a* the slope starts at about 0 MSV (ignoring the dip at zero MSV for the moment which is due to some internal noise). This indicates that the delay line is a quarter period of the nominal data rate. Unfortunately, this causes a degradation in performance at higher MSV. In *Figure 18b* the slope starts at about +3%–4% MSV, and so the delay line is about 3%–4% short at a nominal data rate. This is an optimal delay line length to maximize performance over a ±6%–8% lock range. In *Figure 18c* the delay line is about 3%–4% too long and so there is quite a bit of degradation in window margin in the +MSV portion of the curve.

A final observation is that the wider the bandwidth the lower the window margin, assuming other things like data rate remain constant.

Another interesting fact to note is the type of errors that occur when the bit shift/MSV exceeds the PLL performance. Generally to the left or right of the curve (extreme MSV) the controller will give "Address Mark not Found" errors which is an indication of the PLLs' inability to lock properly. Errors for bit shift that exceeds the curve but for MSV within the PLL's lock range are generally CRC errors, although at very high bit shift a mixture of CRC and Address Mark Errors are expected.

5.3 DP8473 Filter Switching Design Considerations

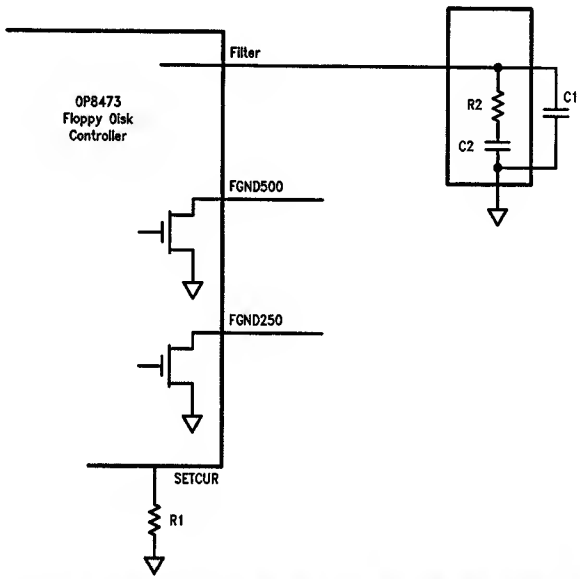
Due to the desire to handle multiple data rates, the DP8473 incorporates on-chip data rate selection logic, and also filter switching logic. In this section we will discuss how this logic works and how to design a set of filters to maximize performance at various combinations of data rates.

Designing with a Single Filter

Previous design examples have dealt with optimization of a single filter at a single data rate. If the DP8473 is to be used at one data rate, the circuit connection is shown in *Figure 19*, and its design is straight forward. It is possible to use a single filter and obtain a reasonable performance at two data rates (i.e., 250 Kb/s and 500 Kb/s or 500 Kb/s and 1 Mb/s). This can be accomplished since the loop bandwidth is scaled by the PLL's divider. Since the divider value increases by a factor of two when going from the high to the low data rate, the bandwidth scales by:

$$\omega_{n250} = \frac{\omega_{n500}}{\sqrt{2}}$$

This scaling is probably not enough to optimize the lower data rate and the damping is affected too, but the single filter approach can still provide acceptable performance in many instances.



TL/F/9419–26

FIGURE 19. DP8473 Filter Configuration for a Single Data Rate Filter, May Be Used as Compromise Filter for Any Two Data Rates

Designing for 250K/300K/500K MFM

This next filter configuration ellows fewer compromises (end hence better performance) when using all 3 PCAT data rates. This configuration is shown in *Figure 20.* This circuit uses two pairs of R's and C's that compose two independent filters. One filter is connected to the FGND250, and the other to FGND500.

To implement the design of *Figure 20*, first design single optimum filters at 250 Kb/s, 300 Kb/s, and 500 Kb/s. Use a single pump resistor for all data rates. Verify and tweek the performance of these filters individually. The 500 Kb/s filter can be directly used. The 250 Kb/s and 300 Kb/s individual filter values need to be compromised into a single R end C filter. C_1 should be $1/20^{th}$ of C_2, and if desired C_3 can be added with e value that is $1/20^{th}$ of C_4.

Designing for 1.0 Mb/s and a 2nd Data Rate

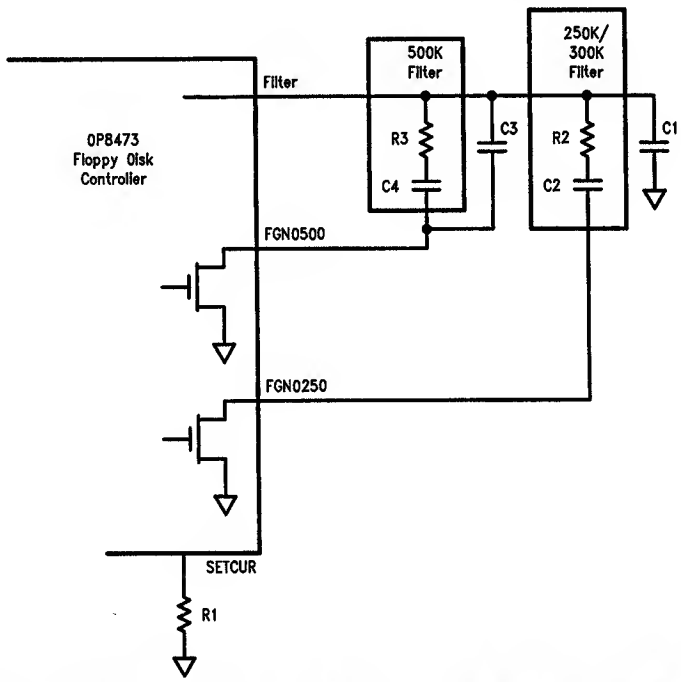
By adding an additional capacitor to *Figure 19* 1.0 Mb/s data rate can be supported, as shown in *Figure 21.* In this figure a single damping resistor is used, but depending on the chosen data rate, one or both capacitors is selected. This configuretion allows the bandwidth to be adjusted more flexibly when designing for the varlous data rates. The design process is, however, a little more complex.

To implement the design of *Figure 21*, first design single optimum filters et 1.0 Mb/s and 500 Kb/s (or 250 Kb/s). Use e single pump resistor for all data rates. Verify and tweak the performance of these filters individually. Using these values, choose e value for R_2 (damping resistor) thet is a good compromise for ell deta retes. Next choose C_2 to be the optimum velue from the initial individual design verification of the 1.0 Mb/s design. Next choose C_3 such that the sum of C_2 end C_3 equals the value for the optimum 500 Kb/s (or 250 Kb/s) design. Finelly, choose C_1 to be $1/20^{th}$ of C_3.

Designing for All Possible Data Rates

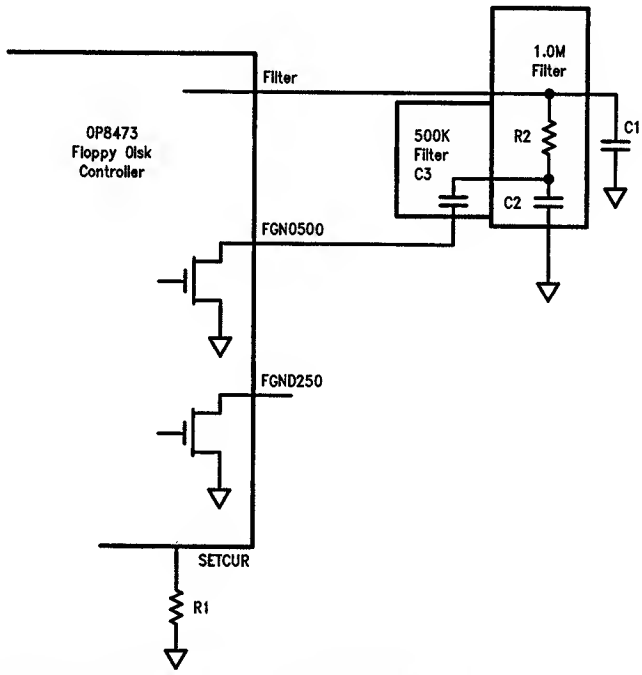
To support ell possible data rates the simplest circuit configuration is one similar to *Figure 20*, but with an additional capacitor selected for the 1 Mb/s data rete.

To implement the design of *Figure 22*, first design single optimum filters at 250 Kb/s, 500 Kb/s, (300 Kb/s also if needed), and 1 Mb/s. Use e single pump resistor for ell data rates. Verify end tweak the performance of these filters individually. Using all of these values, choose a value for R_2 (damping resistor) thet is a good compromise for ell deta rates. Next choose C_2 to be the optimum value from the initial individual design verificetion for 1 Mb/s date rete filter. Next choose C_3 from the 500 Kb/s initiel design. C_3 should be chosen such that C_2+C_3 equals the optimum 500 Kb/s filter value. Then the 250 Kb/s (and 300 Kb/s if used) filter capacitor, C_4, must be chosen in a simlier manner. C_4 should be chosen such that $C_4 + C_2$ equals the optimum 250 Kb/s filter capacitor value, or if using 300 Kb/s it must equal the best compromise 250/300 Kb/s filter capacitor. C_1 should be chosen to be $1/20^{th}$ of C_2.



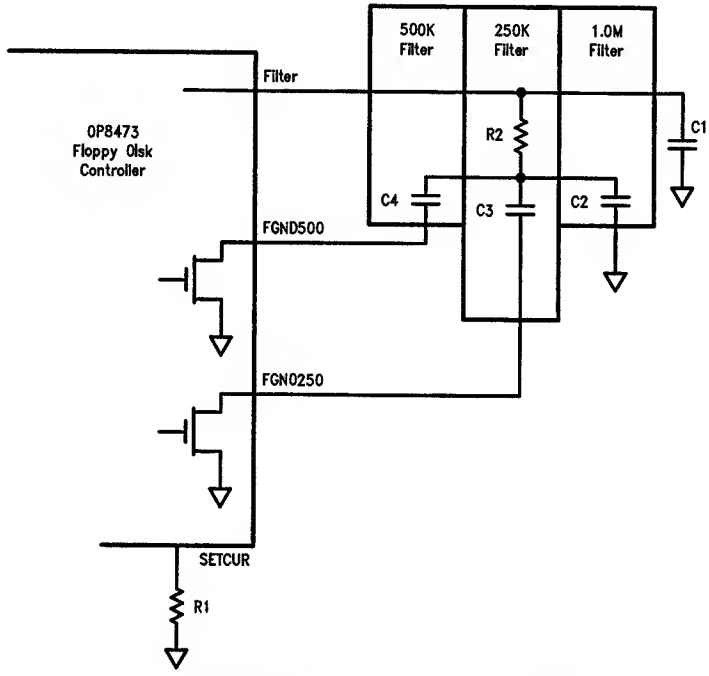
TL/F/9419–27

FIGURE 20. DP8473 Filter Configuration for Optimum 250/300/500 Kb/s (2 Filter) Design



TL/F/9419–28

FIGURE 21. DP8473 Filter Configuration for a Slight Tradeoff Filter Design at 1 Mb and 500 Kb/s



TL/F/9419–29

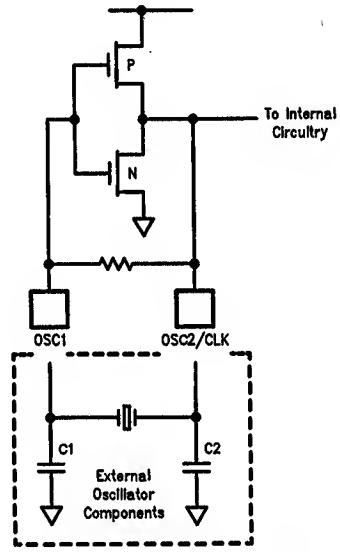
FIGURE 22. DP8473 Filter Configuration for All Data Rates

The previous filter does, however, have some minor performance tradeoffs if all data rates are to be implemented. If the best performance is desired, then the configuration shown in *Figure 23* should be used. In this figure, 3 individual filters are used for each of the data rates. The 1.0 Mb/s filter must be switched via some external circuitry, labeled "ground switch" in the figure. The circuit should enable this filter only when 1 Mb/s data is used, and this filter should be disabled when any other data rate is needed. The circuit to accomplish this could be as simple as an open collector gate derived from the RPM/LC pin, or an alternative that uses no additional hardware is to use an unused drive select output. If the latter option is chosen, then software will have to select the 1 Mb/s filter prior to using this data rate by enabling this bit in the Drive Control Register.

The design of this filter network is very straightforward. Simply design and optimize each filter individually, and use these filter values directly. (Again if 300 Kb/s is also used the 250 Kb/s filter used will be a compromise between the optimal 250 Kb/s and 300 Kb/s filters derived individually.)

5.4 DP8473 Oscillator Design

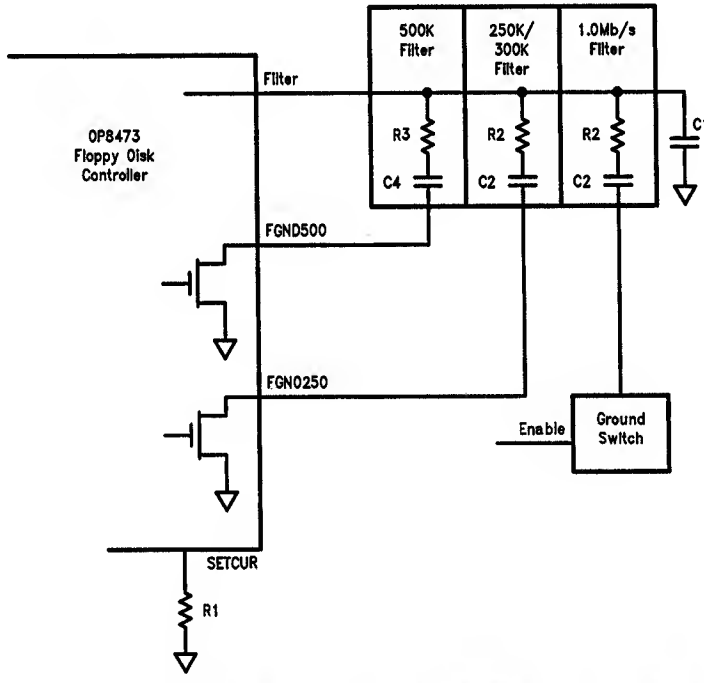
Figure 24 shows the schematic of the crystal oscillator used on the floppy disk controller. This circuit consists of a simple inverter whose impedance has been optimized for use as an oscillator. The inverter is biased into its linear operating region by a high value (>1 MΩ) resistor that is in parallel with the crystal. This biasing allows the inverter to operate as a simple inverting linear gain element.



TL/F/9419–31

FIGURE 24. Simplified Schematic of Oscillator Circuit for DP8473

The DP8473 oscillator is intended to be used with a fundamental mode parallel resonant crystal. The only external components required are the crystal and two external capacitors. These capacitors are usually very small (picofarads).



TL/F/9419–30

FIGURE 23. Filter Configuration for Optimal Filter Design at All Data Rates

TABLE II. Important Parameters for Crystal Selection

Parameter	DP8473
Crystal Frequency	24 MHz
Oscillatory Mode	Fundamental
Oscillator Resonance	Parallel
Accuracy	<0.5%
Series Resistance	<100Ω
Shunt Capacitance	<7 pF
External Parallel Capacitors (include parasitics)	10 pF

Table II shows the important parameters to check for when selecting a crystal to use with the DP8473. While the recommended resonance mode is parallel, a series resonant crystal can be used. It will just oscillate in parallel mode 30 ppm–300 ppm from its ideal frequency.

If an external oscillator circuit is used, it must have a duty cycle of at least 40%–60%, and minimum input levels of 2.4V and 0.4V. The controller should be configured so that the clock is input into the OSC2 pin, and OSC1 is tied to ground.

5.5 Trimming for Perfection

The integrated data separator was designed to achieve excellent performance. However, product, temperature, and power supply variations can degrade performance somewhat. This can lead up to a 10% variation in window margin performance. While this is still exceptional for any analog design, it is possible to trim out this variability.

The two major factors that contribute to data separator performance degradation are: 1) Loop Gain variation; 2) ¼ period delay line length variation.

Trimming the Loop Gain

The loop gain variation can be trimmed by replacing the pump resistor, R_1, with a variable resistor. This resistor should be trimmed based on the ideal lock range of the PLL desired. For example, if a ±6% lock range is desired, then during final board product test, a tester can be used to measure the total lock range and R_1 can be adjusted larger to reduce lock range, or smaller to increase it.

Trimming the Quarter Period Delay Line

The perceived length of the quarter period delay line can be modified by causing a static phase error in the loop to compensate for the quarter period delay line's error. This is accomplished by placing a pull up or pull down resistor on the filter pin. This resistor can be adjusted by measuring the Static Window margin for both an early and late single bit shift. Based on these measurements the delay line can effectively be adjusted by changing the value of the filter pull up/down resistor.

5.6 Initially Unlocked Model

This section is provided in order to complete the full discussion of the theoretical operation of a data separator. It is useful to discuss how the controller locks back to the crystal/clock reference when it needs to. This operation is taken care of by the controller so that the user need not concern himself with the design aspects of this section. However, if the user desires a more complete understanding of the entire lock process that the data separator goes through, this section is presented.

Another model is used to analyze the behavior of the PLL in an unlocked state. It is assumed in this model that the loop is not locked, and the VCO frequency is different from the input frequency. This model can be used to evaluate how the PLL re-locks to the reference clock after reading bad data and being thrown off frequency.

The first operation of the PLL is to frequency lock, so for this model each block is described as a function in terms of frequency, not phase. An equation can be derived for the frequency error that is similar to equation (2) for the phase error:

$$K_e = \frac{\Omega_e}{\Omega_1} = \frac{1}{[1 + K_{DF} K'_{VCO} F(s)]} \tag{26}$$

In the initially unlocked model, the phase detector has a key role. The phase detector compares the VCO output with the input signal. If the VCO output rising edge is leading the input signal, a pump-down signal is generated from this edge of the VCO to the next rising edge of the input signal. If the VCO is lagging the input signal, a pump-up signal is generated from the edge of the input signal to the rising edge of the VCO.

There is no overshoot of the VCO frequency. Only one type of pump signal is generated, up or down, to bring the VCO frequency toward the input frequency. For example, if the VCO frequency is higher than the input frequency, only pump-down signals are generated. It can also be seen that the larger the frequency difference between the VCO and the input, the longer the pump pulses become. The average current flowing from the charge pump is roughly proportional to the frequency difference of the signals at the input of the phase (and now also frequency) detector. The phase detector gain is:

$$K_{DF} = \frac{I_{PUMP}}{\Delta\omega} \tag{27}$$

for $\omega_2 < 2\omega_1$, where $\Delta\omega = \omega_2 - \omega_1$, and

$$K_{DF} = -I_{PUMP} \tag{28}$$

for $\omega_2 > 2\omega_1$. Therefore, if ω_2 is not too far from ω_1, the expression (8) can be written for our PLL as:

$$K_e(s) = \frac{\frac{s\Delta\omega\, C_2}{K'_{VCO}\, I_{PUMP}}}{1 + s\left(R_2 C_2 + \frac{\Delta\omega\, C_2}{K'_{VCO}\, I_{PUMP}}\right)} \tag{29}$$

This expression will allow the calculation of the time that the loop requires to lock back to the reference after a read operation goes through a bad data field or write splice.

Acquisition to the Crystal

After the completion of a read attempt, it is important to ensure that under the worst case conditions the PLL will properly re-lock itself to the reference clock. In order to achieve the required performance during the acquisition to the data stream, the PLL must have reached the lock to the crystal before it is allowed to lock back to the data.

If the PLL attempts to lock to a write splice the VCO may be pulled way off frequency. To prevent this, read gate should be deasserted as soon as a wrong or bad data field is detected. This will prevent the VCO frequency from being pulled too far away. The DP8473 read algorithm has been optimized to prevent this.

If the PLL is locked to the data stream, when read gate is deasserted, the lock mechanism to lock back to the reference frequency is quite similar to locking to the data. If the frequency of the VCO has been swept way off frequency because of a bad data field, noise, write splice, or missing data, the unlocked PLL model must be used. Since when locking to the crystal the phase-frequency comparison is always enabled, the phase detector acts as a frequency discriminator. Switching to the crystal imposes a frequency step to the PLL. It can be demonstrated that the frequency error generated is an exponential function of time, going to 0 with the time constant of:

$$T_P = R_2C_2 + \frac{C_2\Delta\omega R_2 N}{K_{PLL}} \qquad (30)$$

Thus T_P can be assumed to be the worst case acquisition time to the reference clock.

Example 3: From the previous example 1, we have chosen the values for the components of: $C_2 = 0.039\ \mu F$, $R_2 = 535\Omega$. We would like to find the worst case time required to re-lock to the crystal. Assume that the maximum frequency range of the VCO is ±30%.

If the VCO is pulled 30% off center, then:

$$\Delta\omega = 2\pi(500\text{ kHz})\ (0.30) = 942\text{ Krad/sec}$$

Using equation (10) for example 1, we can obtain:

$$T_P = (0.039\ \mu F)(545\Omega) + \frac{(0.039\mu)(942K)(5.6K\Omega)(8)}{(75\text{ Mrad})}$$

$$T_P = 41\ \mu s$$

This is about 4 byte times. Thus, read gate must be deasserted for this length of time before re-asserted to assure that the PLL has relocked to the reference. Most floppy controllers de-assert read gate much longer than this, and the DP8473 deasserts its internal read gate for 6 bytes.

Bibliography

Gardner, Floyd M. PhD., *Phase Locked Techniques*, 2nd edition. New York: John Wiley and Sons, 1979

Best, Roland E., *Phase Locked Loops*, New York: McGraw-Hill, 1984

DP8470 Phase Locked Loop Data Sheet, National Semiconductor Corp. 1986,7

DP8472 Floppy Disk Controller PLUS Data Sheet, National Semiconductor Corp. 1986,7

DP8473 Floppy Disk Controller PLUS/2 Data Sheet, National Semiconductor Corp. 1987

Design Guide for DP8473 in a PC-AT

National Semiconductor
Application Note 631
Robert Lutz

OVERVIEW

When designing a floppy interface circuit for a PC-AT in the past, there was very little flexibility given to the design engineer. The NEC765, which was designed into the original IBM PC, was the only floppy controller available that would guarantee compatibility. Compatibility is extremely important in a PC design.

There were many design issues that had to be resolved when using the NEC765 to produce a fully functional floppy controller interface. A data separator had to be selected or designed. A write precompensation circuit needed to be included. A whole score of miscellaneous logic had to be designed to handle all of the unique PC functions that the NEC765 does not handle itself.

The DP8473 from National Semiconductor was developed to eliminate all of these design problems. All of the extra functions and logic normally required for an XT or an AT design were included inside the chip. Even an analog data separator, which is classically the hardest function to design, has been integrated into the DP8473.

Compatibility has been completely retained. The DP8473 is software compatible with the NEC765A. We have not found any current software available for the PC that fails to work properly with the DP8473. This includes software running under DOS and OS/2.

This application note will discuss some of the issues involved in a floppy controller design with the DP8473. Even though on the surface, there may not seem to be many options when designing a floppy controller for a PC, there really are quite a few. Some of these options include: signal swapping in the floppy cable, different types of floppy drives, and data separator filter selection.

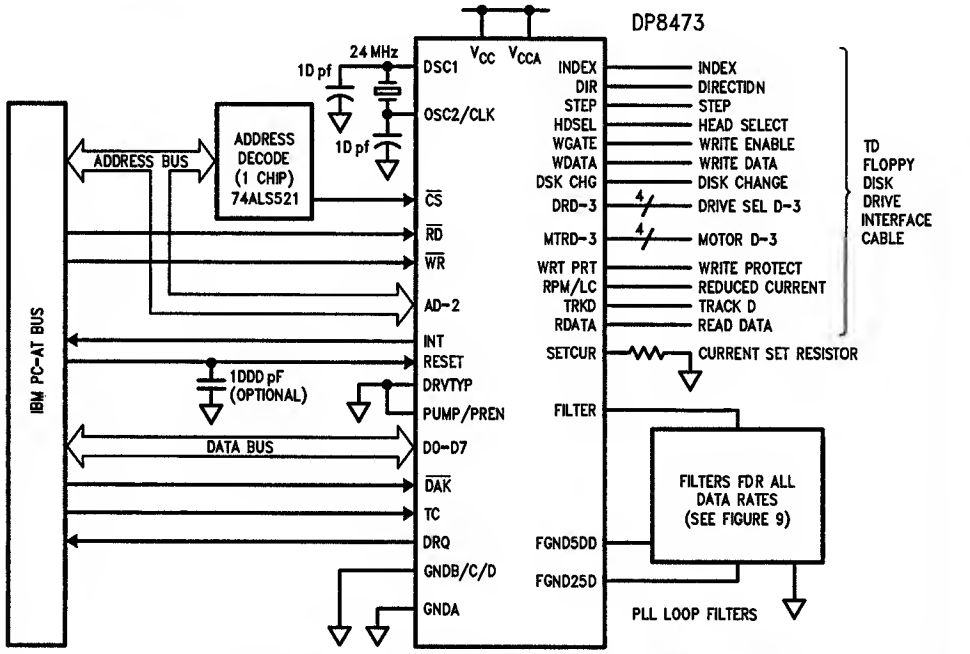


FIGURE 1. Schematic of Typical DP8473 Floppy Controller Design

HARDWARE ENVIRONMENT

A typical floppy controller design with the DP8473 will look something like the schematic shown in *Figure 1*. You may be surprised that the entire schematic for the floppy controller design fits on less than one page. Especially if you consider that the schematic for a similar function in the IBM PC-XT technical reference manual takes four full pages.

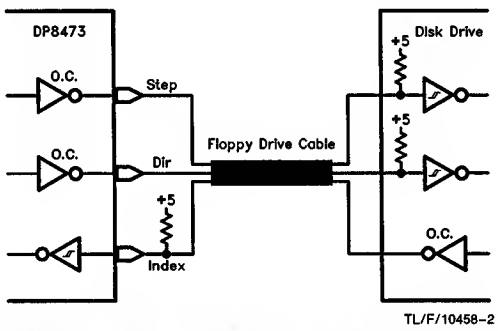
The heart of the design is, of course, the DP8473. Most of the interface pins to and from the DP8473 go directly to the peripheral bus or the disk drive cable without additional logic or buffering.

DRIVE CABLE INTERFACE

The DP8473 disk interface signals connect directly to the drive cable. Most disk drives terminate the drive cable with resistors. Termination is required because the output buffers of the floppy controller are open-collector.

Terminated signals are used because historically, relatively long cables have been used to connect the floppy controller to the disk drives. The cable termination will decrease the amount of crosstalk and noise on the drive cable signals.

A typical disk interface circuit consists of an open-collector output buffer at the signal source, and a termination resistor and a Schmitt input buffer at the signal destination. For example, the STEP output pin on the DP8473 is an open-collector output buffer that is capable of sinking up to 48 mA (See Note 1). If the output is off, the buffer is disabled, and the termination resistor on the disk drive will pull the signal high. If the STEP output is on, the DP8473 buffer will pull the signal low. An example of the cable termination logic is shown in *Figure 2*.



TL/F/10458–2

Note 1: The DP8473 actually contains open-drain output buffers due to its CMOS design. The end result is the same as TTL open-collector buffers.

FIGURE 2. Example of Buffers and Terminators Used for Floppy Drive Interface

The termination resistors used with 5.25″ drives or 8″ drives typically have a value of 150Ω. With this resistor value, the output buffers must be capable of sinking about 35 mA each in order to pull the drive signal to a logic low level. The DP8473 is able to sink this current without external buffers.

The termination resistors used with 3.5″ drives are often 1 kΩ. 1 kΩ termination resistors are also sometimes found on low power 5.25″ drives. Drive manufacturers have recognized that the floppy interface cable used in a PC is relatively short. Also, the drives are installed in the same grounded enclosure as the PC and the floppy controller. This reduces the amount of noise introduced on the floppy interface cable.

If both 3.5″ drives and 5.25″ drives are to be used in an application, the termination resistors used with the DP8473 must be chosen carefully. The termination resistor value used with the DP8473 must be the larger of the termination resistors used on the drives. For example, if the 5.25″ drive has 150Ω termination resistors and the 3.5″ drive has 1 kΩ termination resistors, the termination resistors used on the inputs to the DP8473 should be 1 kΩ.

The termination resistors for the inputs to the DP8473 should be placed near the DP8473. The termination resistors for the outputs for the DP8473 to the disk drive are contained in the disk drive itself.

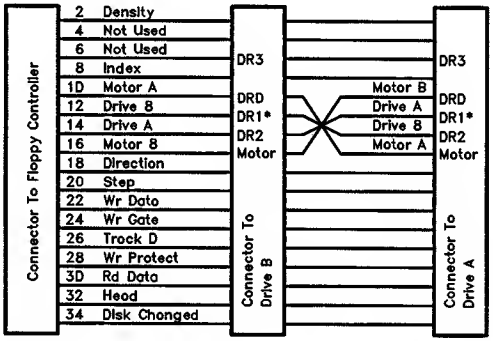
Additional disk interface buffering is not normally required when using the DP8473. It can sink up to 48 mA for each disk output signal. This is more than enough capacity for a typical floppy design.

DRIVE CONFIGURATION

A PC-XT can typically interface to up to four disk drives. A PC-AT, however, is usually limited to two disk drives. The two drive limitation is due to the ROM BIOS used in most AT's. More than two drives can be used if special software drivers are written.

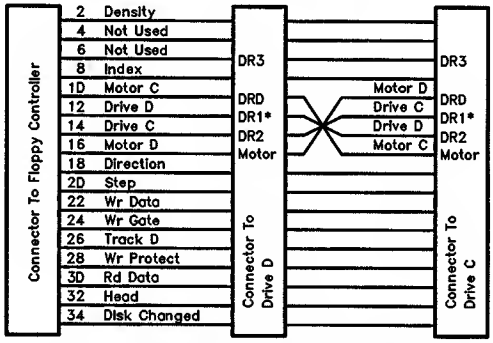
The connection between the floppy drives and the floppy controller in a PC is slightly different than the SA450 standard used in non-PC's. The advantage to the method used in a PC is that each disk drive installed in the PC can be configured identically. Even the Drive Select strap is the same. Each drive is configured as drive 1 (or B). Even if four drives are all connected, they are each strapped as drive 1.

The trick used to accomplish this feat is cable wire swapping. The drive cable is cut up and wires are moved around. The cable swapping re-routes the four DRIVE select signals (DR0, DR1, DR2, DR3) to the DR1 signal of each individual drive. For example, DR0 is routed to drive A's DR1 input. DR1 is routed to Drive B's DR1 input, and so on. In a similar manner, the cable swapping also re-routes the four MOTOR signals (MTR0, MTR1, MTR2, MTR3) to the MOTOR signal of each drive. *Figure 3a* demonstrates how the cable is configured for two floppy disk drives. A second cable would be used if more than two drives are required as shown in *Figure 3b*.



TL/F/10458–3

FIGURE 3a. Cable Swapping Used for Drives A and B



TL/F/10458–4

FIGURE 3b. Cable Swapping Used for Drives C and D

Note: The asterisk (*) next to DR1 indicates that this drive is strapped as Drive 1. In the PC, all drives are strapped as Drive 1.

If more than one disk drive is attached to the floppy controller, there may be more than one drive terminated. This may cause a current overloading problem. The controller may not be able to drive an active signal low.

It is easy to prevent current overloading in a two disk drive PC. Simply make sure that only one drive is terminated. Ideally the terminated drive should be the drive at the end of the drive cable, although it could be either drive.

If both drives are terminated, the output buffers will be driving too much current. The system will be out of specification. This is a common situation and is largely ignored by PC manufacturers. The output buffers can usually handle the additional load.

It three or four drives are to be used, things become more complex. For example, if four 150Ω terminated drives are all attached to the DP8473, the DP8473 will have to sink 139 mA for each drive interface signal. The DP8473 is guaranteed to sink up to 48 mA. Therefore, this configuration would not work without additional buffering. Please refer to the How to Calculate the Maximum Current Required for Output Buffers section for a description on current calculation.

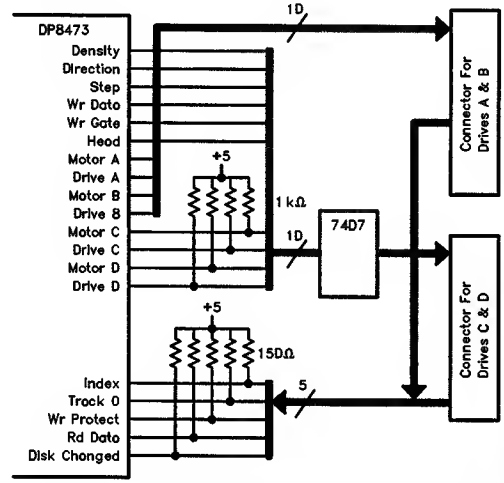
There are three techniques that can be used to prevent overloading the output buffers:

Technique 1:

Using larger termination resistors can reduce the load on the DP8473 to acceptable levels. If 1 kΩ resistors can be used instead of 150Ω resistors. In the worst case where all four disk drives are terminated, only 21 mA will be generated instead of 140 mA. This is well within the specification of the DP8473. 1 kΩ resistors are commonly used with 3.5" drives.

Technique 2:

The most direct technique that can be used is simply adding additional buffers for the extra disk drives. Drives A and B can be driven directly by the DP8473. The outputs to drives C and D can pass through an open-collector buffer such as the 7407. This is shown in *Figure 4*. 1k pullup resistors are required for some of the DP8473 outputs because they are not terminated by drives A or B.

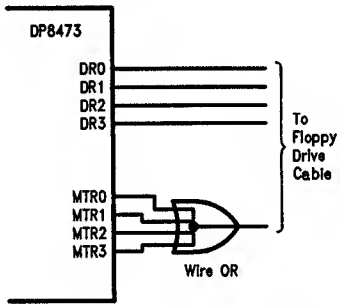


TL/F/10458–5

FIGURE 4. Extra Buffers Required for Four Drive System

Technique 3:

Daisy chain the floppy drives with the controller on one end of the drive interface cable, and one terminated drive at the other end. One to three additional non-terminated drives can be added in the middle as shown in *Figure 6*. With this technique, the four Motor signals from the DP8473 should be wire-ORed together as shown in *Figure 5*. Each drive must be strapped for the proper drive select (0–3).



TL/F/10458–6

FIGURE 5. Wire OR Required for Daisy Chain Connection

How to Calculate the Maximum Current Required for Output Buffers.

Since a floppy controller design may not work correctly due to current overloading, it is important to understand exactly how to calculate the maximum current required by the floppy controller for each output signal to the disk drive. This is largely determined by the termination resistors used by the disk drives. A formula that can be used is:

$$\frac{(V_{CC}) + (\text{Max } V_{CC} \text{ Variation}) - (V_{OL(max)})}{(\text{Termination Resistor}) \bullet \frac{1}{N} \bullet (1 - \text{Resistor Accuracy})}$$

$V_{CC} = 5.0$

Max V_{CC} Variation = Power supply variation (0.5V)

$V_{OL(Max)}$ = Maximum active low output voltage of buffer (0.8V)

Termination Resistor = Termination on disk drive

N = Number of terminated disk drives

Resistor Accuracy = Accuracy of termination resistors (10% or 0.10)

Example 1:

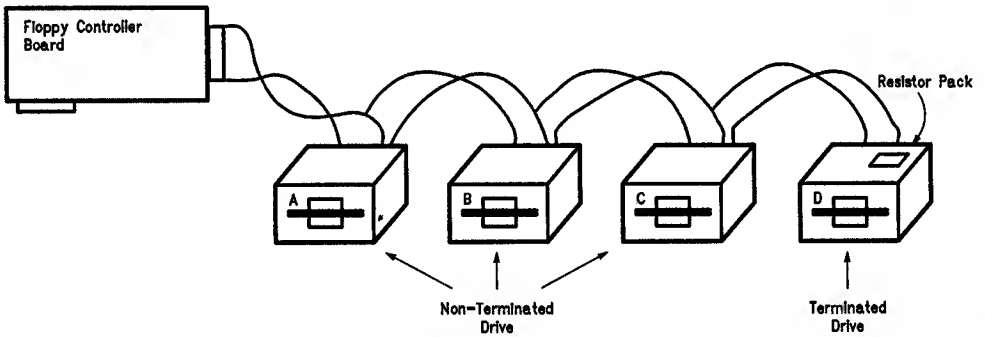
One terminated drive with 150 termination resistors.

$$\frac{5.0 + 0.5 - 0.8}{150 \bullet 1 \bullet 0.9} = 34.8 \text{ mA}$$

Example 2:

Four terminated drives with 1k termination resistors.

$$\frac{5.0 + 0.5 - 0.8}{1000 \bullet \frac{1}{4} \bullet 0.9} = 20.9 \text{ mA}$$



TL/F/10458–7

FIGURE 6. Daisy Chain of Four Drives

DRIVE TYPES

There are many types of disk drives that can be connected to the DP8473 floppy disk controller. The DP8473 is compatible with 8", 5.25", and 3.5" floppy disk drives, although 8" drives are rarely used today.

Other types of peripherals may be connected to the floppy controller as well. A streeming tape drive that is used to back up the hard disk is often connected to the floppy controller. A tape drive of this type is very specialized. It has been designed to look like a floppy disk drive from an electrical interface point of view. It does not perform exactly like a disk drive, however. Special software is usually required to meke it work correctly. The STEP signal is often used to issue commands to the tape drive. For example, to rewind the tape, four step pulses may need to be issued. To start a read, six step pulses might be issued.

All of these different drive types have one thing in common, a similar electrical interface. This allows them all to be connected to a common drive interface cable. For example, the READ DATA signal on pin 30 of the floppy interface cable is the MFM encoded serial stream of data from the disk. The INDEX signal on pin 8 of the cable identifies the beginning of a track.

However, there are some minor differences between drive tapes that must be considered. The DENSITY signal is a good example of a difference. This signal is ective for *high* density transfers on a dual density 3.5" drives. But, this signal is active for *low* density transfers on a dual density 5.25" drive. This difference makes the floppy system design more complex when 5.25" dual density end 3.5" dual density drives are both used in the same PC.

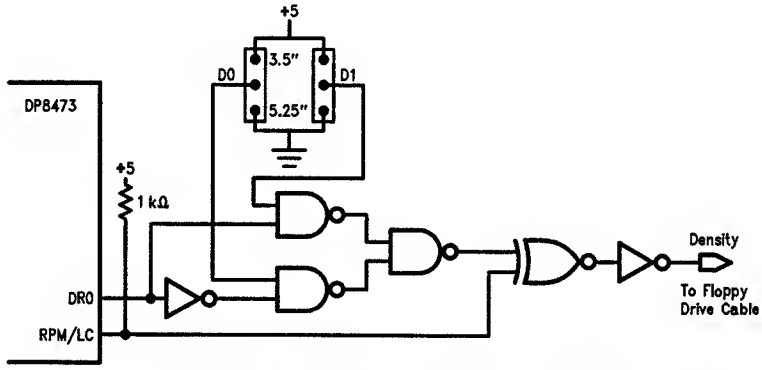
The DP8473 has a signal called RPM/LC that normally connects to the Density or Low Current input on a dual density 5.25" drive. If a 3.5" drive is used, the RPM/LC output should be inverted.

A design such as the one shown in *Figure 7* could be used to create the proper DENSITY signal for both 5.25" and 3.5" drives. The jumpers end logic allow the user to select between drive types for each individual drive.

Another solution is simply to use a 3.5" drive that contains a built-in jumper to vary the polarity of the DENSITY signal directly on the drive. This option eliminates the need for external logic and jumpers.

Another signal that is drive type dependent is DISK CHANGED. This signal exists on low and dual density 3.5" drives and elso on dual density 5.25" drives. It does not exist on low density 5.25" drives. If a low density 5.25" drive is to be used, the DISK CHANGED signal should be held active (low level). It may be held active by the drive by itself. If not, a pull-down resistor could be used to activate the non-driven signal.

One thing to consider while choosing drive types is media compatibility. Of course, you can't put a 3.5" disk in a 5.25" drive. But, there are more subtle incompatibilities even within similar media types. For example, a low density 5.25" disk written to in low density mode by a dual density drive cannot be read reliably by a low density drive. Table I lists the compatabilities between different drives and media types.



TL/F/10458–8

FIGURE 7. Density Select Logic for 5.25" or 3.5" Drives

TABLE I. Drive and Media Compatibility

Drive Type	Mode	Media Type			
		3.5″	3.5″ HD	5.25″ DD	5.25″ HD
3.5″ LD	720k	R/W			
3.5″ LD,HD	720k 1.44M	R/W	R/W		
5.25″ LD	360k			R/W	
5.25″ LD,HD	360k 1.2M			R Only	R/W

Notes:
LD = Low Density
HD = High Density
R/W = Readable & Writable
R Only = Readable Only

DATA RATES

Different drive types may use different data rates. The data rate is specified by the number of bits that are transferred in a second. For example, 250 kb/s is translated to 250 thousand bits per second.

The data rate used by a disk drive is determined by the electronics of the drive and the specifications of the media. Low Density media require data to be transferred at 250 kb/s. High Density media is twice as fast at 500 kb/s.

There is a complication with Low Density 5.25″ media in a Dual Density drive. The 5.25″ Dual Density drive spins the disk faster (360 RPM) than a Low Density drive (300 RPM). When a Low Density disk is read from a Dual Density drive, the data rate will be 300 kb/s instead of 250 kb/s because of the rotational speed difference.

The DP8473 can operate at all the data rates required for a PC. This includes 250, 300, and 500 kb/s. In addition, the DP8473 can operate at 1 Mb/s. This high data rate is starting to appear in both floppy disks and streaming tape drives.

μP INTERFACE

It is hard to imagine how the interface between the DP8473 and the μP bus could be made any simpler than it is. Only one interface function is not integrated in the DP8473. That function is address decoding. The typical μP connections are shown in *Figure 8*.

The DP8473 requires a CS (chip select) enable signal that is generated elsewhere. Typically, this could be generated by an ALS521 8-bit comparator or a similar circuit. Address decoding of the three least significant bits of the address is performed by the DP8473. The AEN (address enable) signal from the μP bus should be included in the CS logic. This will prevent DMA transfers from generating a CS.

The eight bit data bus from the DP8473 connects directly to the data bus of the μP. Bus transceivers are not required because 12 mA buffers are built into the DP8473.

The IOR (I/O read), IOW (I/O write), RESET, TC (terminal count), DRQ (DMA request), and IRQ (interrupt request) signals can be connected directly to the DP8473. No additional buffering or gating is required. The logic required to TRI-STATE® the DRQ and INT pins is integrated in the DP8473.

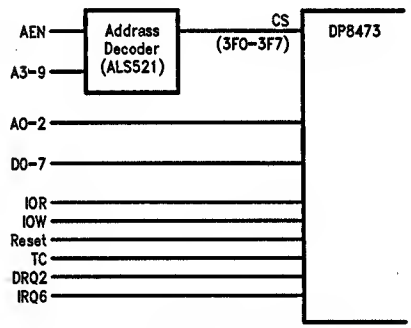


FIGURE 8. μP Interface to DP8473

FILTER SELECTION

The internal data separator in the DP8473 requires an external filter to operate correctly. This filter is part of an analog Phase-Locked Loop (PLL) that is used by the data separator integrated into the DP8473. This is commonly referred to as an analog data separator due to the analog nature of the PLL's filter and Voltage Controlled Oscillator (V_{CO}).

As the floppy controller changes from one data rate to another, the filter used by the PLL must change also. This is done automatically in the DP8473. Two or three filters are connected externally to different pins of the DP8473. The correct filter is selected and activated by the DP8473 itself.

The filter selection is performed by grounding or forcing TRI-STATE the appropriate filters. For example, a two-filter arrangement is shown in *Figure 9*. If 500 kb/s is selected, the FGND500 pin is grounded and the FGND250 pin is at TRI-STATE. From the filter pin's point of view, this appears as filter F2 in parallel with capacitor C1. F1 is electrically out of the picture. When 250 kb/s is selected, it is the opposite. The FGND500 pin is at TRI-STATE and the FGND250 pin is grounded. Since 300 kb/s data rate is close to 250 kb/s, the same filter is used for both data rates.

The two-filter arrangement shown in *Figure 9* would be used in most PC applications. It supports 250, 300, and 500 kb/s data rate. Other filter combinations may be used for specialized applications. These filter combinations are described in the DP8473 data sheet.

DATA SEPARATOR PERFORMANCE

One of the most important features of the DP8473 is the high level of data separator performance. This performance translates directly to reduced disk I/O errors. There is quite a bit of information that can be discussed on data separator performance. A lot of this information is presented in an application note titled "Floppy Disk Data Separator Design Guide for the DP8473, AN-505".

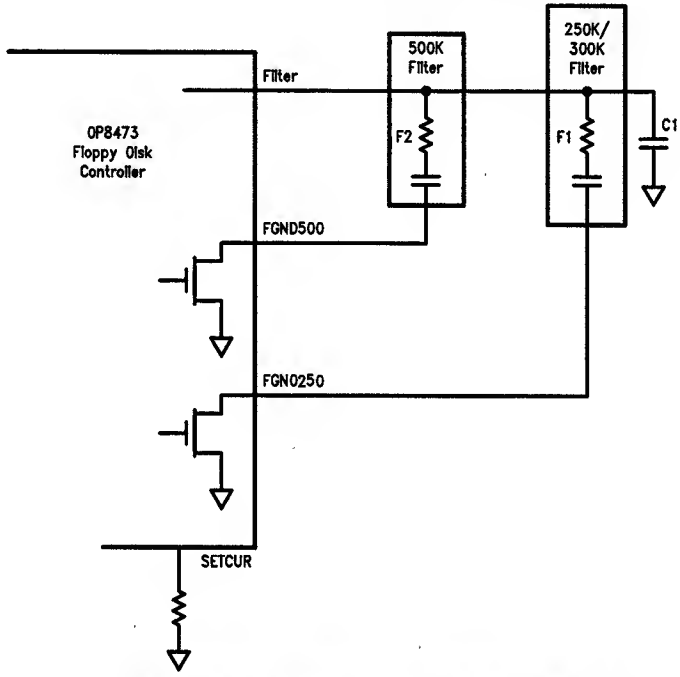
It might seem desirable to specify the maximum error rate of the DP8473. This could be expressed in terms of the number of bits that can be read correctly before an error occurs (10^{12}, for example). This is not a practical parameter to specify, however. One problem is the amount of time this type of measurement would take. A test of 10^{12} could take well over 400 days to measure.

Another problem is defining the test conditions. Is the test performed under ideal disk conditions? Are bits jittered or is motor speed variation simulated? There are so many variables in this type of a test that it would be difficult for two different people to produce the same results. In addition, the error rate is related to other factors beside the DP8473 such as the disk drive or the floppy media.

There are many different types of measurements that can be made with a data separator. Most of these measurements indicate how well only one particular section of the data separator performs. For example, the gain of the VCO (Voltage Controlled Oscillator) or the accuracy of the ¼ period delay line. They don't, however, give a good indication of how well the data separator will perform under real-life conditions.

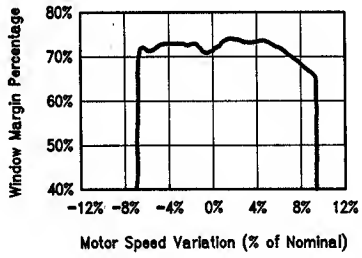
There is one measurement that produces meaningful data. This measurement is called "Dynamic Window Margin". Dynamic window margin attempts to indicate total data separator performance under real-life conditions. It measures how much bit jitter the data separator can handle while reading a worst case data pattern (DB6 hex) with a drive that has a motor speed of varying accuracy. The data is jittered in a manner similar to real world jitter with a reverse write precompensation pattern. The measurement is taken at the worst point over a motor speed variation of ±6%. An example of a dynamic window margin measurement is shown in *Figure 10.*

The typical dynamic window margin is specified in the DP8473 data sheet. National Semiconductor has made measurements of the dynamic window margin of many data separators, and the results show that no other data separator performs as well as the DP8473s.



TL/F/10458–10

FIGURE 9. Typical Filter Connection for 250, 300, and 500 kb/s



TL/F/10458–11

FIGURE 10. Typical Dynamic Window Margin for DP8473

LAYOUT AND COMPONENT CONSIDERATIONS

The DP8473 contains a combination of digital and analog circuitry. Because of the analog nature of the data separator, some precautions should be taken when designing a board with the DP8473.

A good DP8473 board layout design will distinguish between analog and digital, V_{CC} and Ground. The supply pins used by the digital circuitry are labeled V_{CC}, GNDB, GNDC, and GNDD. The supply pins used for the analog data separator are labeled V_{CC}A and GNDA. Standard digital decoupling techniques should be used with the digital supply pins. This typically involves 0.1 μF capacitors connected between V_{CC} and GND.

The analog supply pins require a bit more consideration than the digital supply pins. Any noise or ripple on the analog supplies will degrade the data separator performance. It is recommended to minimize this noise as much as possible. Less than 50 mV noise would be good.

There are many methods that can be used to minimize noise on the analog supply pins. One of the best methods is a 5.0V voltage regulator dedicated to the analog section. This guarantees a very clean signal. The voltage regulator can be driven by the 12V power supply.

Another method is to place the DP8473 on the board close to the entry point of the power supply. At the very least, separate supply lines should be dedicated from the power supply entry point to the DP8473 V_{CC}A and GNDA.

In addition to the analog supply, any noise or crosstalk introduced to the external filters will adversely effect the performance of the data separator. The data separator filters should be positioned as close as possible to the DP8473. A ground plane surrounding the filters would also be advisable. The resistor attached to the SETCUR pin should also be close to the DP8473.

If a 24 MHz crystal is used, it should be placed close to the DP8473 as well as the 10 pF capacitor attached to both sides of the crystal.

The component types and tolerances may also effect the data separator performance. The accuracy of the resistor attached to the SETCUR pin is important. It should have a 1% tolerance rating. A 5% could be used, but the accuracy may effect the data separator performance.

The capacitor in series with the resistor attached to the FILTER pin is also critical. It should have a 5% tolerance. The series resistor in the same network is not as critical as the SETCUR resistor. Therefore, a normal 5% tolerance can be used here.

Finally, the capacitor in parallel with the filter network can be rated as low as 10%–20%. It does not effect the data separator very much.

The component tolerances mentioned here are only recommendations. The DP8473 will work properly with a wide range of filter values. These recommendations should be followed if data separator performance is an important issue in a particular design (which is normally true).

TROUBLE-SHOOTING

If the floppy controller does not appear to operate correctly, there are some key areas that can be looked at for the source of the problem.

Drive's in Use light remains on at all times.

- Drive Interface cable plugged in backward.
- Drive Interface signals not properly routed.

DOS returns a "Not ready error reading drive X".

This error can be caused by many different problems. At this point it would be best to run the "Floppy Demo Program" as described later in this section.

DOS Directory command returns an old directory.

- Disk Changed signal improperly routed.

POST produces a "601" error while booting.

- Drive Interface cable unplugged.
- Hardware problem with μP interface.

PC locks up while booting.

- Hardware problem with μP interface.

Parity Error.

- DMA interface problem.

Many advanced diagnostics may be used while running the "Floppy Demo Program" which is available from National Semiconductor. This program allows you to issue individual floppy commands such as Read Data, Format Track, and Seek. The Result Phase of these commands can be analyzed to help determine where a problem exists. The following list describes some likely sources of problems based on what is observed with the "Floppy Demo Program".

"Missing Address Mark in Address Field" error.

This indicates that the floppy controller could not find any valid data on the track being read. No sectors could be found.

- Track has not been formatted. Could be a blank disk. The controller does recognize Index Pulses, however. This indicates that the drive cable interface is at least partially intact.
- Read Data drive interface signal not properly routed.
- General data separator problem. See the data separator discussion later.

"Did not receive interrupt" error.

This is usually accompanied with all FF's in the Result Phase of the command. This indicates that the controller could not read anything from the disk drive and Index pulses were not seen. Be sure to reset the floppy controller after this error.

- Disk not inserted in drive or drive door open.
- Wrong drive selected in command.
- Drive Interface cable unplugged.

MSR (Main Status Register) does not return to 80 (hex) after a reset.

- Floppy controller not inserted in socket properly.
- Hardware problem with address decode (CS) or μP interface signals.
- Crystal or external clock not operating properly.
- Floppy controller is bad.

"No Data" error or "CRC error".

- Bad disk media.
- General data separator problem.

Fewer bytes read than requested or error in Result Phase.

- Noise on Reset pin. Insert capacitor between Reset and digital GND as specified in the data sheet.

Long term read produces some errors.

- General data separator problem.

Many of the problems described above refer to a general data separator problem. There are many things that can be looked at for data separator problems.

- Filter wired incorrectly.
- Incorrect filter component values.
- Filter layed-out poorly.
- Too much noise on analog $V_{CC}A$ or GNDA.

FLOPPY CONTROLLER DESIGN WITH NEC765A

In order to appreciate the amount of circuitry integrated into the DP8473, it may be useful to analyze a typical floppy controller design for the PC-AT using the standard NEC785A floppy disk controller. To simplify the analysis, only the block diagram will be presented. The actual schematic would require many pages. The block diagram is shown in *Figure 11*.

The NEC765A was developed many years ago originally for 8" floppy disk drives. It became very popular because it was designed into the original IBM PC. The NEC765A performs many functions, but there are also many functions that it does not perform.

A data separator isolates the individual pulses read from the disk drive and allows the floppy controller to distinguish between MFM clock pulses and data pulses. It typically incorporates an analog PLL. An analog data separator design could require as many as 10 chips to design. A single chip discrete digital data separator could also be used, although the performance will not be as good.

The write precompensation circuit shifts the MFM encoded data as it is being written to the disk. This shifting compensates for known bit shifts that will occur due to the magnetic influences of the individual bits recorded on the disk. This is typically designed with a shift register and a multiplexer.

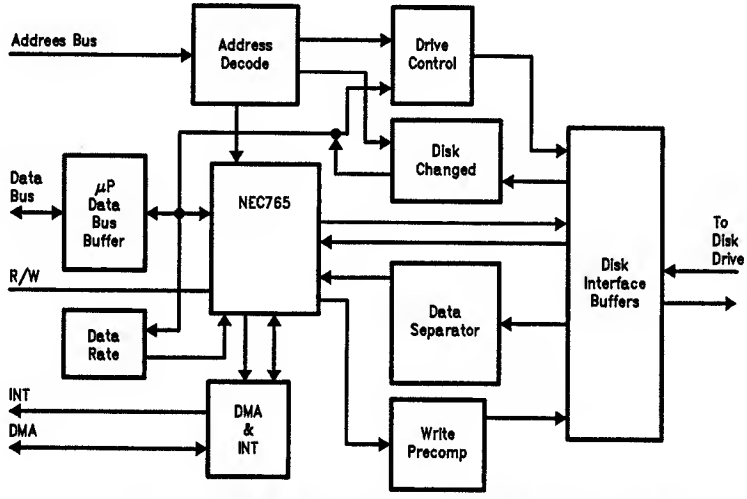
The NEC765 cannot interface directly to the Drive Interface cable. Separate 48 mA buffers are required for each output signal. This requires three 7406's. Also, the inputs from the disk drive required Schmitt inverters.

Additional buffering is also required for the μP data bus.

The PC-AT requires that the Disk Changed signal be read from a particular port. This involves address decoding and buffering.

The only method available to vary the data rate used by the NEC785A is by altering the input clock frequency to the chip. This must be done with a complex divider circuit that generates the different frequencies required for 250, 300, and 500 kb/s data rates.

Due to timing incompatabilities in the NEC765A, the Drive Select and Motor On signals must be generated by an external port that is controlled by software. This port also controls the software reset and DMA and INT enable circuitry.



TL/F/10458–12

FIGURE 11. Block Diagram of Floppy Controller Design with NEC765

DMA transfers must be slowed down due to handshaking problems with the NEC765A. This delay is performed with an external shift register.

This entire design easily requires at least a couple of dozen chips. The amount of board space used is quite large. A considerable amount of current is also consumed. This compares to the DP8473 solution which only requires two chips. It is easy to see that the DP8473 solution is much more economical.

CONCLUSION

This design guide was created to answer the most common questions encountered while designing with the DP8473. Any new design can be based on the information given in this guide. A two drive system can be created or more drives can be added if required. A variety of disk drives may be used including 3.5" drives.

The address decoding is the only function not integrated into the DP8473. However, the integrated data separator requires external filtering which should be carefully layed-out on the board.

If problems arise, there are many items that can be looked at to help identify where the problem exists. It may be useful to obtain a floppy controller diagnostic program similar to the "Floppy Demo Program" available from a National Semiconductor sales office.

If more information is desired concerning the performance of the data separator or the trade-offs of designing a custom filter for the PLL, please read the application note titled "Floppy Disk Data Separator Design Guide for the DP8473", AN-505.